Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Refining a 3D GAN model for predicting tumoral growth on t2-Flair brain MRIs

Author:
Mathieu Charbonnel

Supervisor:
Matthew Williams
Co-Supervisor:
Elsa Angelini

Submitted in partial fulfillment of the requirements for the MSc degree in Artificial Intelligence and Machine Learning of Imperial College London

September 2020

Abstract

This project aims to predict brain glioma growth from a baseline MRI. It builds upon proof-of-concept results using a extended version of Pix2Pix using pairs of 2D FLAIR brain MRI tagged with time-lapse information. We intend to investigate the following sophistications:
(1) Explore Pix2Pix on 3D MRI brain tumour images
(2) Explore the recent Edge-aware Ea-GAN architecture to improve prediction quality.
(3) Add a Time Predictor Model to improve growth prediction;
(4) Try using several input images from the same patient to predict an ulterior MRI;
The project builds on Andreas Zinonos' work from last year as an Msc student.

Acknowledgments

Foremost, I would like to express my sincere gratitude to my supervisor Dr. Matthew Williams and co-supervisor Dr. Elsa Angelini for offering me to work under their supervision on this very exciting project. Both were extremely supportive and interested in distinct aspects of the topic, which made interactions with them challenging and stimulating.
A special thanks to Dr. Matthew Williams for providing the medical data that I had the privilege to work with, and communicating his motivation and help at each weekly meeting to see the project move forward.
A special thanks as well to Elsa Angelini for letting me use a GPU machine located in the ITMAT Data Science group and giving me a lot of freedom overall. Indeed, when it comes to coding, this project was the hardest for me to date, and being able to solve issues at my own pace at the beginning of the project was very fortunate.
Finally, while I did not talk to her in person, I would like to thank Kammy Pike for preparing the private dataset and performing segmentation masking for some of the images.

Contents

1 Introduction **1**

2 Background research **2**
2.1 Imaging Protocols . . . 2
2.2 Computer Vision . . . 4
2.2.1 MRI image calibration . . . 4
2.2.2 Registration . . . 4
2.2.3 Labelling and tumour segmentation . . . 4
2.2.4 Atlas of functional resectability of gliomas . . . 5
2.2.5 Glioma growth prediction . . . 5
2.3 Generative Models uses . . . 6
2.3.1 Data Augmentation . . . 6
2.3.2 Imaging Modality Translation . . . 6
2.3.3 tumour Evolution Prediction . . . 6
2.4 GANs . . . 7
2.4.1 Variants and architectures . . . 8
2.4.2 Loss metrics . . . 13
2.4.3 Evaluation metrics . . . 15

3 Starting Point and Project Direction **19**
3.1 Introduction: Andreas' Work . . . 19
3.2 Dataset . . . 19
3.3 Project Direction . . . 19
3.3.1 Preprocessing . . . 20
3.3.2 Edge aware GAN models . . . 20
3.3.3 Edge aware GAN with Time Predictor . . . 21
3.3.4 DM-gEA-GAN . . . 22

4 Implementation **23**
4.1 Preprocessing . . . 23
4.1.1 Brain Masking . . . 23
4.1.2 Coregistration . . . 24
4.1.3 Upsampling . . . 24
4.1.4 Pairing . . . 24
4.1.5 Split between training set and test set . . . 25
4.1.6 Re-Scaling . . . 25

4.1.7 Data Augmentation 25
4.1.8 For Time Predictor: Difference Map 26
4.1.9 Alternative pairing for Difference Map gEA-GAN 26
4.1.10 Alternative Preprocessing with tumour Masking 26
4.2 Models and Architectures 27
4.2.1 Introduction 27
4.2.2 Hyperparameters 27
4.2.3 3D Pix2pix 29
4.2.4 Generative Edge Aware GAN and Discriminative Edge Aware GAN 30
4.2.5 Time Predictor and TPN-Ea-GAN 31
4.2.6 DM-gEA-GAN 32

5 Experiments **33**
5.1 3D pix2pix 33
5.1.1 Parameters 33
5.1.2 Losses 33
5.1.3 Brain Images from the test set 34
5.2 64-gEA-GAN 36
5.2.1 Parameters 36
5.2.2 Losses 36
5.2.3 Brain Images from the test set 37
5.3 gEA-GAN 38
5.3.1 Parameters 38
5.3.2 Losses 38
5.3.3 Brain Images from the test set 39
5.4 dEA-GAN 41
5.4.1 Parameters 41
5.4.2 Losses 41
5.4.3 Brain Images from the test set 42
5.5 Time predictor 44
5.5.1 Parameters 44
5.5.2 Loss 44
5.6 TPN-gEA-GAN 45
5.6.1 Parameters 45
5.6.2 Losses 45
5.6.3 Brain Images from the test set 46
5.7 Visual global comparison 48
5.8 DM-gEA-GAN 51
5.8.1 Parameters 51
5.8.2 Loss 51
5.8.3 Visuals 52

6 Evaluation Metric **53**
6.1 Final choice of metrics . . . 53
6.1.1 Expectations . . . 53
6.1.2 Algorithms and Formulas . . . 54
6.1.3 Implementation and combination of the metric scores . . . 56
6.2 Experimental Results . . . 57
6.2.1 All scores comparison . . . 57
6.2.2 Total score comparison . . . 57
6.2.3 Experimental Results Analysis . . . 57

7 Discussion **58**
7.1 On brain masking and coregistration . . . 58
7.2 On predicting 128x128x128 images 58
7.3 ...With a 36-patients dataset . . . 59
7.4 Edge Aware GAN for 2D slices . . . 59
7.5 Ideas for new 3D models . . . 60
7.5.1 Multimodal GAN . . . 60
7.5.2 Multi Scale Model . . . 60

8 Conclusion **61**

9 Appendix **63**
9.1 Ressources . . . 63
9.1.1 Machine . . . 63
9.1.2 Tools and Frameworks . . . 63
9.2 Index of work credit . . . 65
9.2.1 Performed by Kammy Pike . . . 65
9.2.2 Performed by Elsa Angelini . . . 65
9.2.3 Performed by Andreas Zinonos . . . 65
9.2.4 Performed by Biting Yu . . . 65
9.2.5 Performed by myself . . . 66
9.3 Legal and Ethics . . . 67
9.4 User Guide-Typical Session . . . 69
9.4.1 Preprocessing . . . 69
9.4.2 Training and Testing . . . 69
9.4.3 Evaluation . . . 70

Chapter 1

Introduction

Brain gliomas are the most common primary malignant brain tumors in adults. They are classified in categories based on their cellular morphology, and can evolve in various ways and show different responses to treatments. The exact cause of gliomas cannot be defined with certainty but hereditary disorders, diets, radiation, infection with some viruses are believed to potentially cooperate in their development. While low grade gliomas can grow very slowly and be followed without treatment, high-grade gliomas are vascular tumors and can infiltrate diffusely. These low high grade gliomas almost always grow back after surgical excision.
Treatment also vary but often rely on combinations of surgery, radiation therapy and chemotherapy. Whatever the best treatment method is, getting precise images of the tumour location and shape is vital. Computer Vision and Deep Learning algorithms have been used a lot on brain gliomas MR images since 2010. They mostly focus on brain tumour segmentation, synthesis of new images, for instance to transfer known images into images of different modalities (Kaiser and Albarqouni (2019)). These new images can also serve as data augmentation for existing algorithms.
The Generative Adversarial Network architecture is a example of an algorithm able to produce ”realistic” tumour MRI. This model trains two networks. The generator network learns to generate ”realistic” output while the discriminator learns to output the probability of an image being real or fake. This model is at the core of the project and will be referred to very often in this report.
Andreas, as well as I, focused our study on a particular implementation of GAN: Pix2Pix. He applied it to brain slices from our private dataset to perform glioma growth prediction between two longitudinal scans. We applied a 3D version of it to full 3D t2-Flair MRI from this same dataset. In the background research chapter we will detail what exactly Pix2Pix applied to glioma growth prediction is, along with important details that are related to it. In the Implementation chapter we will describe very precisely the steps required to obtain our brain tumour 3D predictions. In the Experiments Chapter we will describe the exact parameters and models we used in several experiments, as well as the visuals we obtained.In the Evaluation Metric section we will detail our final metrics and present the results obtained with the different architectures and parameters we tried. The discussions chapter will enable us to mention all the issues and thoughts we had along the way that could apply to other projects. We will also use that section to discuss potential future work.

Chapter 2

Background research

2.1 Imaging Protocols for brain gliomas

Magnetic resonance imaging (MRI) is widely used to assess brain gliomas. Computed tomography (CT) is also available and its image acquisition is relatively quick, however only MRI can show anatomical details and normal brain structures as well as tumor-infiltrated areas (Pope and Brandal (2018)). Moreover a wide array of MR pulse sequences can be tailored to detect a variety of intracranial aberrations. MRI sequences commonly used for evaluation of intracranial malignancy include T1-weighted, T2-weighted , Fluid Attenuated Inversion Recovery, T2-W gradient echo and post-contrast T1W images.
MRI physics is based on the magnetization properties of protons (MRIBasics[2006]). An external field aligns the proton spins within the tissue being examined. After the alignment is disrupted by Radio Frequency energy, the protons gradually return to there normal spin. As they do, they emit a signal, which is recorded by antennas and then converted to gray scale intensity levels using Fourier Transformation. Contrast between tissues is determined by the rate at which excited atoms return to equilibrium state. Time to Echo (TE) is the time between the delivery of the RF pulse and the receipt of the echo signal. Multiple pulses are delivered to encode spatial location. Repetition Time (TR) is the amount of time between successive pulse sequences.
Tissues are characterized by two different relaxation times: the longitudinal relaxation time (T1) and the transverse relaxation time (T2). The T1-weighted and T2-weighted images try to capture the information carried by those two relaxation times. FLAIR sequences are similar to T2 except that the TE and TR times are longer, as a result, the edema become bright wile liquids such as normal CSF fluid are made black.

A contrast agent can allow for the visualization of blood and detection of areas where the blood-brain barrier is compromised.
Yet, MRI and CT are not always enough to distinguish various glioma grades and underlying histology. Gliomas can also be difficult to distinguish from other intracranial mass lesions.

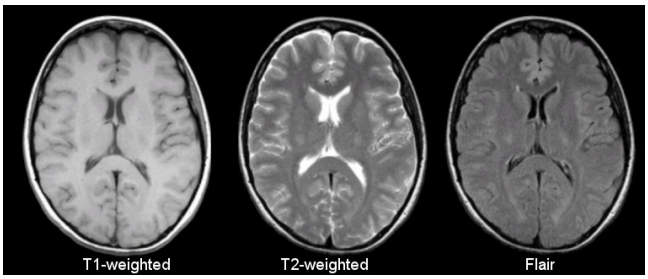


Figure 2.1: Comparison of T1 vs. T2 vs. Flair on a healthy brain from MRIBasics[2006] website

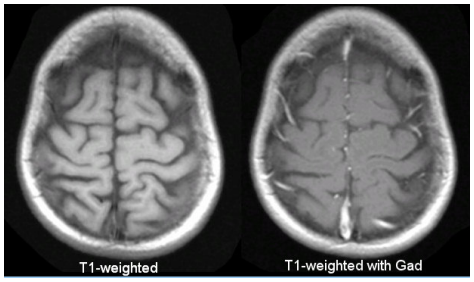


Figure 2.2: Comparison of T1 vs. T1 enhanced with Gadolinium on a healthy brain from MRIBasics[2006] website

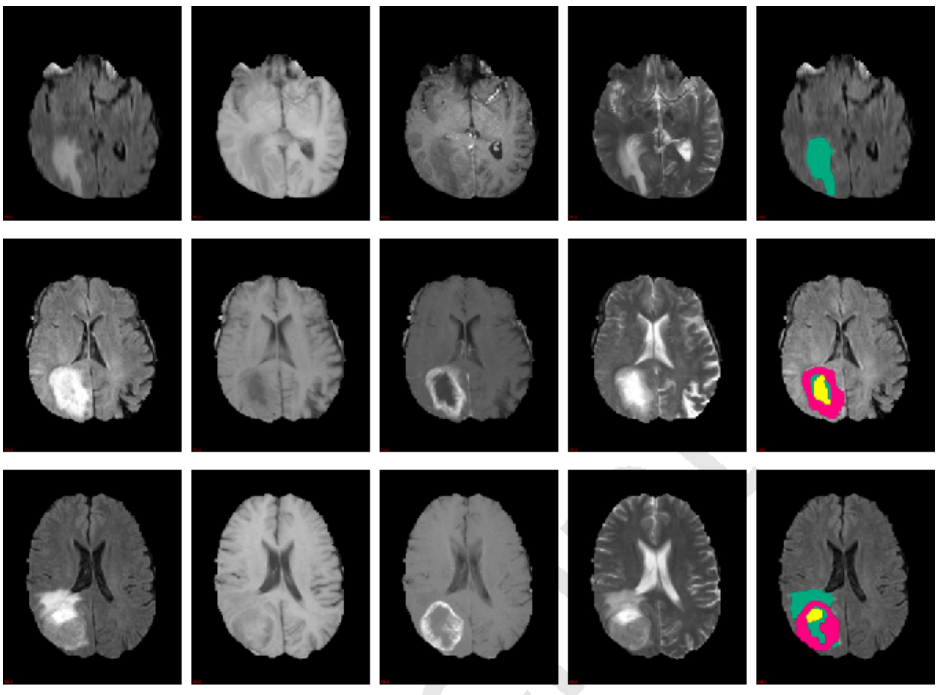


Figure 2.3: Axial slice for three different high-grade glioma, from Li et al. (2016). From left to right: FLAIR image, T1 image, T1C image, T2 image, and the segmentation results (yellow = necrosis, green = edema, rosy = enhancing tumor)

2.2 Brain glioma MRI and Computer Vision

2.2.1 MRI image calibration

Most application using clinical MRI rely heavily on accurate calibration of imaging gradients. This calibration is in practice performed during maintenance sessions by manufacturer. When using MRI performed at different times they do not necessarily follow the same calibration and suffer from large variability. As a result it can be needed to make sure that the intensities of images follow a reference distribution (via histogram matching for example).

2.2.2 Registration

Image registration is the process of spatially aligning images with a source image. The goal is to find a geometrical transformation which maps corresponding structures in the target and the source image. Registration is necessary in many cases. First, aligning brain MRI images of the same patient taken at different times can greatly improve performance of algorithms that try to predict future images, and make overall progression of the tumour more readable. But Registration must also be used when applying generic simulations to specific patients.
There are different ways of performing registration, usually it is simplified to rigid deformation when the source and target images are from the same patient. Indeed the image was probably not taken from the very same angle but it is unlikely that the brain of the patient was warped or underwent nonrigid transformation. Obviously the presence of a tumour (which evolves) makes registration harder. Defining a model for tumour displacement or alleviating the rigid condition are ways to deal with it (Angelini et al. (2007))

2.2.3 Labelling and tumour segmentation

Segmentation and accurate estimation of the volumes of brain subcomponents is used for monitoring progression of a tumor, radiotherapy planning, outcome assessment and follow-up studies. When done by human experts, this is a complex task as tumors appear under different forms, and different MRI have to be used to classify the tumour sub-components (necrosis, enhancing tumor, non-enhancing tumor- Agravat and Raval (2020)). The time required plus the potential human mistakes makes it a very interesting task to automate if possible. Convolutional models have recently achieved good results on these kinds of tasks. Kamnitsas et al. (2016) presents an architecture designed for stroke lesions segmentations, the Deep Medic. In our work we will use manual masks for brain and glioma in order to help our GAN to predict tumoral growth. But if such automation is efficient enough, this expensive (in time) masking could be done systematically for baseline images.

2.2.4 Atlas of functional resectability of gliomas

Another difficulty that appears when treating Brain gliomas is the dynamic capability (plasticity) of the brain to reorganize itself (Ius et al. (2011)). If an area of the brain is damaged, a neuronal aggregate can adopt its function. Recent advances have increased the comprehension and prediction of dynamic interactions between tumour and functional cortical sites. Efficient surgery requires to be able to predict the resectability of different areas. The task of mapping these is not obvious due to to the functional reorganization but needs to be done in order for the surgeon not to remove essential cortical areas (which would lead to permanent deficit), in opposition to ”modulatory” areas that can be functionally compensated. In Ius et al. (2011), an atlas from a homogenous group of 58 patients with lesion in eloquent regions was built from FLAIR MRI. Cortical and subcortical mapping could be performed using direct electrical stimulation and led to such an atlas.

2.2.5 Glioma growth prediction

MRI can be used to gain information on the state of a patient's brain glioma (via segmentation etc), but also to predict growth evolution. Reaction-diffusion models are commonly used in literature for this task. They can incorporate more or less complex information like different tissue types, geometry of the brain and the directions of white matter fiber tracts. However all brain are different, and the adaptation of a general model to a specific patient case is a problem in itself. In Konukoglu et al. (2010) this is dealt with by estimating parameters that govern tumour evolution for every patient time series of medical images. Promising results on 2 real cases seem to imply that these parameters capture well the specificity of each patient.

2.3 Generative models serving brain glioma MRI study

2.3.1 Data Augmentation

Having enough data is vital when training machine learning algorithms to perform non-obvious tasks. Medical imaging brings another issues: abnormal findings are less common (by definition) and are often what the scientist is interested in. So it is harder to get a good variability as well as class balance. And these are necessary to train robust classifiers (Shin et al. (2018)).
While traditional augmentation like cropping or translation/rotation of the image can help, they do not really increase the diversity of the data in terms of lesion (shape and size). The idea in Shin et al. (2018) is to use image synthesis as a potential solution to this. The generative model here is a GAN conditioned on a segmentation mask of brain anatomy and tumour and producing multi-parametric MRI of corresponding abnormal brains. From BRATS (public dataset for Multimodal Brain Tumor Segmentation Challenge) T1 MRI, the model infers a Brain Atlas, which is merged with a given tumour label, and from this merged image, the T1, T1c, T2 and T2 FLAIR are inferred. That way it is possible to generate images of a brain with different tumour sizes in different locations. This data also has the advantage of being anonymous, and can be shared more freely that real images from an hospital.

2.3.2 Imaging Modality Translation

Application of Imaging Modality Translation are numerous and generative models have recently learned to translate image domains, for instance photos from day to night, objects from drawings to pictures and so on. It is possible to translate an MRI into a CT. That is exactly what is done in Kaiser and Albarqouni (2019). It assumes that MRI are more informative in a way and carry all the information that CT have, in another form. And MRI are supposed to be a lot more harmless than CT. The GAN Pix2Pix architecture used in Kaiser and Albarqouni (2019) proves to be the most efficient in this specific modality translation.

2.3.3 tumour Evolution Prediction

Tumour evolution prediction is a subject that has naturally been researched on. Especially an important medical advance would be to be able to extract parameters that define a given tumor. As a tumour can evolve in a lot of different ways, being able to visually extract these parameters from an MRI would be determinant in predicting its evolution. Such concerns are investigated in papers like Ezhov et al. (2019). This very paper considers reaction-diffusion equations as an efficient model for predicting tumour evolution, and aims at finding its parameters from MRI inputs. Generative models can be used for Evolution prediction. Typically, GANs are used for prediction in different areas including Du et al. (2019) but in the context of brain glioma, it is an original idea. Andreas used a GAN conditioned on an MRI taken at t_0 as well as the timelapse $t_1 - t_0$ to predict an image at t_1. He used T2-FLAIR images,

restricted his work to longitudinal slices of the brain and obtained realistic results. tumour Evolution Prediction is the application of Image Synthesis that we will focus on in this project. Like Andreas, we use a GAN, more precisely a Pix2Pix, but extend his work in various ways.

2.4 GANs

The concept of Generative Adversarial networks was introduced by Goodfellow et al. (2014). The idea is to train a generative model which replicates the distribution of a real dataset. Wen considering images, for instance, the goal is to have a model that produces images that are realistic but also covers the data distribution of the real dataset. Typically the generative model should not produce the same output again and again.
In 2014, generative models already existed. It was possible to give a noise to a neural network and expect it to produce realistic (in regards to a given dataset) output. For that to happen a loss is needed to tell how far the generated images are from the real dataset. This is the piece of information that will be propagated backward in the neural network and lead to a change in its weights. A naive loss could be the euclidean distance. But this favors blurry images as being "not to far" from the real dataset comes down to averaging. Blurry images are not realistic, and carry no information.
So the problem of a good generative model comes down to finding the perfect loss that tells how far a generated image is from the real dataset. The original idea of a GAN is: why not training another network to learn the loss function ? This will be the discriminator. Basically it learns the chance that a given image is fake, and discriminate fake images from real images (Figure 2.4). We end up with a min-max problem where the losses for the generator G and the discriminator D are:

$$\begin{aligned} L(D) &= -\mathbb{E}\left[\log(D(x_r))\right] - \mathbb{E}\left[\log(1 - D(G(z)))\right] \\ L(G) &= \mathbb{E}\left[\log(1 - D(G(z)))\right] \end{aligned} \tag{2.1}$$

where z is a random variable that denotes the noise given as input to the generator, x_g is a random variable that denotes the generated image, x_r is a random variable that denotes a real image and follow the distribution given by the dataset, and finally y is a random variable that denotes the probability that an image is real, estimated by the discriminator.

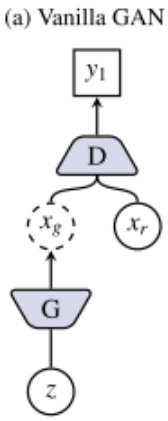


Figure 2.4: Vanilla GAN architecture from Yi et al. (2019)

GANs are very powerful but suffer from some issues. First, speed of learning of discriminator and generator can be very different. The most common case appears when the discriminator learns much faster than the generator. It is then able to tell almost perfectly a real image from a fake one. The gradient for the generator vanishes as it has very little loss difference when producing a better image (the discriminator tells it is fake anyway) and learning takes forever.
Another problem is called ”mode collapse”, which happens when the generator starts producing the same image again and again as the discriminator at first is not able to discriminate it. At some point the collapse point moves as the discriminator starts noticing it is fake but the distribution of generated data stays highly clustered around this point.

Let us go through a few specific architectures that will be used in this project in the context of brain glioma growth prediction.

2.4.1 Variants and architectures

Conditional GANs

Conditional GAN is the first idea mentioned as a possible straightforward extension in Goodfellow et al. (2014). It is based on adding another input to the generator as well as the discriminator: the condition. It is useful if we want the generator to output something of a given category, or given an information. In this report we will use conditional GANs to predict future image of a brain **given** a baseline. This baseline will be the ”condition”.
The only variable that changes from the vanilla GAN is the condidition that acts as another input c (Figure 2.5). It is the new input for the generator and the discriminator.

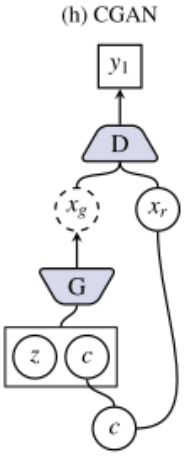


Figure 2.5: Conditional GAN architecture from Yi et al. (2019)

U-net model

GANs training being often unstable and uncontrollable, many variants of the vanilla GAN have been introduced to try to overcome these issues, like VAE-GAN, RS-GAN, T-GAN (Yi et al. (2019)). Here we will quickly consider U-like algorithms as it is the architecture that is used with Pix2Pix which is very efficient in the area of medical imaging.
U-net architecture was introduced by Ronneberger et al. (2015) and its most important operations are up-sampling and skip connection concatenation of the modules between encoder and decoder. More precisely it uses an encoder (left branch of the ”U”) and a decoder (right branch of the ”U”) built from fully convolutional networks with non linearity function between each up-sampling step. Transpose convolution operation produce up-sampled feature maps so that the symmetrical inverse mapping is automatically generated. Then feature maps skipped from the encoding part join the decoding part of U-net (horizontal transfer inside the ”U”).

As a result, the U-net structure can incorporate context information at multiple scales and propagate the abstract layers (bottom of the U) directly to the concrete layers (Top of the ”U”). (Jionghao (2019)).

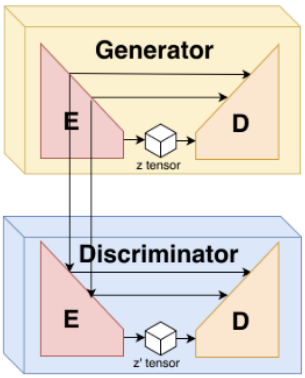


Figure 2.6: U-shape GAN typical structure from Jionghao (2019)

Figure 2.6 shows the general UU-net family applied to GAN. Both the generator G and the discriminator D are U-like models based on an encoder E coupled with a decoder D. The horizontal arrows represent the skip connections between the encoder (left part of the ”U”) and decoder (right part of the ”U”). In this particular case some features maps are also given to the discriminator. z and z' are the latent representations given by the two encoders.

Pix2Pix

Pix2Pix is an established conditional GAN architecture that is used to perform image to image translation tasks. An image is given as an input to the generator, and it is supposed to generate the corresponding image. The discriminator classifies a given pair of images as the real pair or the generated pair. In Pix2Pix, the generator is a convolutional network with U-net architecture. The discriminator is a ”patchGAN network” and takes the concatenated input images. On top of the adversarial loss, a L1 loss term is used between the target image and the generated image. It is added with a coefficient to the generator loss and makes Pix2pix somewhat more supervised. These losses are detailed in the next section.

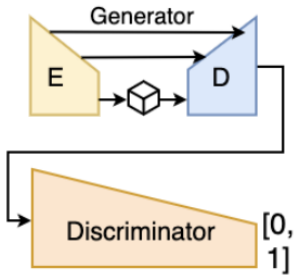


Figure 2.7: Pix2Pix features that special U-net architecture in its generator only, diagram from Jionghao (2019)

Edge Aware GANs

The idea of edge aware generative adversarial networks was introduced in Yu et al. (2019). This very recent architecture is based on the Pix2Pix algorithm but integrates the extraction and analysis of the edge maps (Figure 2.8).
The edge maps are computed using a three-dimensional Sobel edge operator based on three kernels denoted by F_i, F_j, F_k. These kernels are of size $3 \times 3 \times 3$.

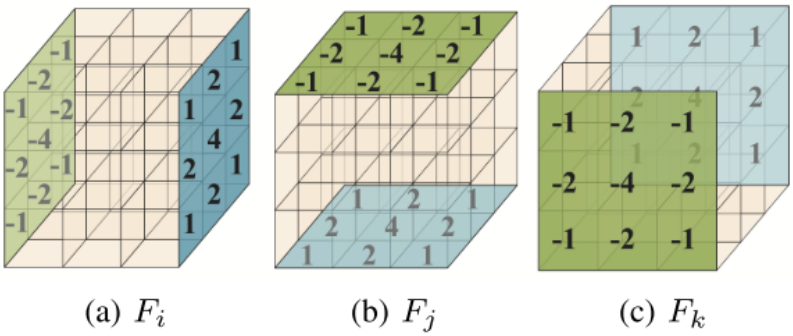


Figure 2.8: Sobel edge kernels from Yu et al. (2019), ”empty cubes” means a value of zero at the corresponding position

$$S(A) = \sqrt{(F_i * A)^2 + (F_j * A)^2 + (F_k * A)^2} \tag{2.2}$$

where A is the image and * is a convolution.

These edge maps S(A) are supposed to contain critical textural information for visual recognition. The study performed in the paper with two architectures (Generator Edge Aware GAN and Discriminator Edge Aware GAN) shows that using edge information enables to boost the quality of generated samples. These models are 3D based, and show one main difference, dEA-GAN is the only one to give this edge information to the discriminator.

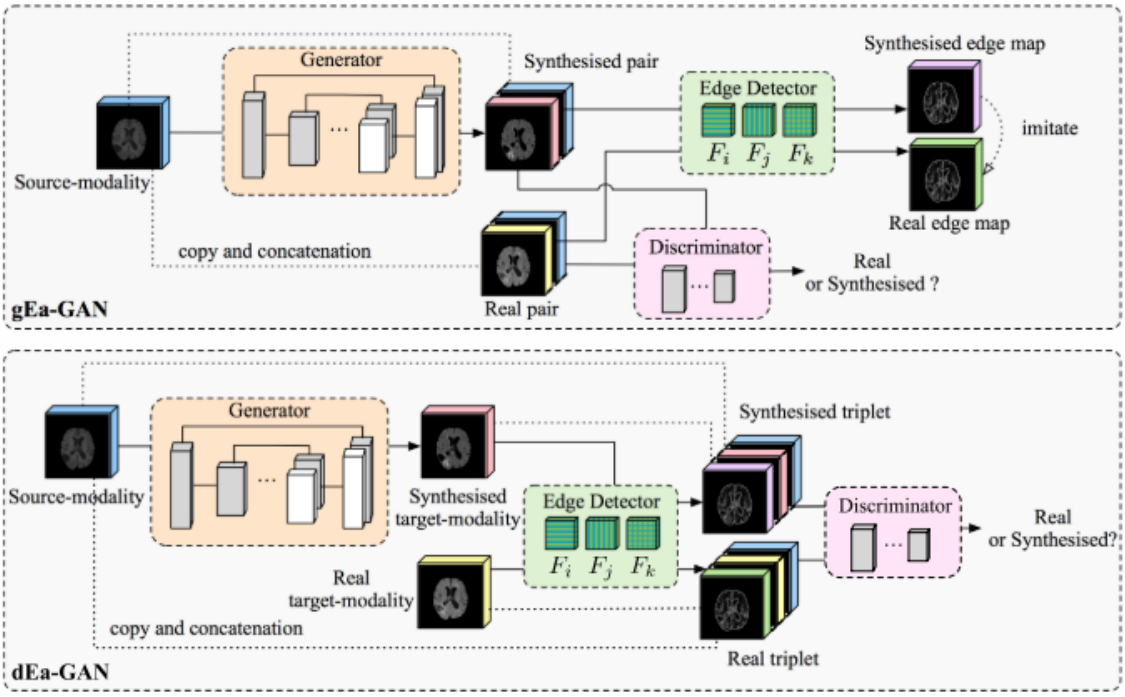


Figure 2.9: Two Edge aware GAN architectures from Yu et al. (2019)

2.4.2 Loss metrics

Adversarial Loss

This loss is what makes GANs a very special and interesting generative model. Instead of trying to compare the generated image with the real one via a distance loss, it uses the loss given by the discriminator which was trained to differentiate such images. In other words instead of tailoring by hand a perfect loss for our generator, the learning discriminator is given this task. As it needs to be trained, the adversarial loss also includes the discriminator loss. At the end, as the discriminator D outputs a probability (between 0 and 1), the GAN (adversarial) loss is written:

$$\mathcal{L}_{GAN} = \min_G \max_D \mathbb{E}_{y\sim p(y)}[\log(D(y))] + E_{z\sim p_z}[\log(1 - D(G(z)))] \tag{2.3}$$

where G represents the generator, D the discriminator, $p(y)$ the real dataset distribution and z the noise used by the generator (Goodfellow et al. (2014))
Conditional GANs take an input (x) as well as a noise signal (z) so the loss is written:

$$\mathcal{L}_{cGAN} = \min_G \max_D \mathbb{E}_{x,y}[\log(D(x, y))] + E_{x,z}[\log(1 - D(x, G(x, z)))] \tag{2.4}$$

Weighted L1 Loss

L1 loss is a very basic loss, computing the absolute pixel-wise distance between the prediction and the real image. "Weighted" means that it is possible to give more weight to some parts of the image. For example more importance can be given to bright areas which are more likely to correspond to the tumour area. The model is told to focus on that part for the prediction. Using an importance factor λ, and denoting by X, Y the predicted and target image while $i \in$[1,N] indexes the pixels, the weighted L1 loss is

$$\mathcal{L}_{wL1}(\lambda) = \mathbb{E}_{(X,Y)}[(1 - \lambda)\sum_{i=1}^{N} |X_i - Y_i| + \lambda \sum_{i=1}^{N} |X_i - Y_i| \mathbb{1}_{X_i > I_{thresh}}] \tag{2.5}$$

where I_{thresh} is the intensity threshold for what is considered as the bright area, and $\mathbb{1}$ is the indicator function.

Pix2Pix

Pix2Pix is the baseline that will be used in this project. Pix2Pix Loss is the sum of a traditional conditional GAN adversarial network with a weighted L1 Loss which seemed to increase its performance. If we denote by μ the relative importance of the weighted L1 loss, the total Pix2Pix loss is:

$$\mathcal{L}_{Pix2Pix} = \mathcal{L}_{cGAN} + \mu \mathcal{L}_{wL1}(\lambda) \tag{2.6}$$

Edge aware model losses

In Yu et al. (2019) two architectures are presented. Both use the three-dimensional Sobel edge operator mentioned on the previous section
In the case of the generator edge aware GAN, a Sobel L1 loss (L1 distance between the edge maps of the predicted and real images) is added to the Pix2Pix loss as follows:

$$\mathcal{L}_{gEA-GAN} = \mathcal{L}_{cGAN} + \mu\mathcal{L}_{wL1}(\lambda) + \lambda_{edge}\mathbb{E}_{(x,y),z}[\|S(y) - S(G(x,z))\|_1] \tag{2.7}$$

where λ_{edge} is a coefficient of importance given to the Sobel L1 distances and S is the Sobel operator.

The dEA-GAN uses this L1 distance between the edge maps in the generator but also adds an input to the discriminator which is the edge map of the predicted/target image. As this architecture is probably the one that will be at the core of this project, let us detail the generator and discriminator losses for the dEA-GAN (Yu et al. (2019)):

$$\begin{aligned} \mathcal{L}^{G}_{dEA-GAN} &= E_{x,z}[\log(1 - D(x, G(x,z), S(G(x,z))))] + \mu\mathcal{L}_{wL1} \\ &\quad + \lambda_{edge}\mathbb{E}_{(x,y),z}[\|S(y) - S(G(x,z))\|_1] \\ \mathcal{L}^{D}_{dEA-GAN} &= -E_{(x,y)}[\log(D(x,y,S(y)))] - E_{x,z}[\log(1 - D(x, G(x,z), S(G(x,z))))] \end{aligned} \tag{2.8}$$

As a result the dEA GAN generator loss is similar to the gEA GAN generator loss with only one notable difference: the discriminator called loss also uses the edge map. The discriminator loss is the conditional adversarial loss with one additional input to the generator: the edge maps of the generated/target images.

Time loss

This loss has been developed in Andreas' project in an attempt to take into account the time difference between the pairs of training MRI for a given patient. In practice he trained a third network named TPN to predict this time difference. On top of being used as an additional loss, the time is also concatenated as a channel in both the Discriminator and Generator. The final architecture of the TPN that Andreas used was inspired from a model used to predict the steering angle of driver-less vehicles given images of the road (Du et al. (2019)). To our knowledge it is an original idea in the context of predicting the growth of a brain glioma..

2.4.3 Evaluation metrics

The adversarial loss which characterizes GANs allows to circumvent the need of a good hand-made loss but there is still the need for a relevant evaluation metric to tell if generated images are good enough. Much of the ”public” work on GANs (e.g. pix2pix with style transfer) is informally evaluated. This is for a reason: finding a good evaluation metric for GANs is actually a very hard problem. Two main things are required from the generated images: being realistic, and fitting the real dataset distribution. Images coming from similar latent values must be close enough, but the overall diversity of the generated dataset needs to be big enough (as mode collapse is not desirable). Many metrics were proposed in the literature. Here we will present a few interesting ones, and give a first idea on how well they could perform on our specific brain glioma growth prediction.

Reconstruction Error

The most obvious way to evaluate our specific model is, given a real baseline as well as a timelapse, to compare the generated MRI (conditioned by this information) with the real image. Reconstruction error is an immediate pixel to pixel comparison between these. It can take into account the absolute intensity difference between corresponding pixels (Mean Average Error) or the squared difference (Mean Square Error). This metric suffers from the same issues as most basic losses in generative models: they favor blurry, ”averaged” images more than realistic ones. Let us review a few others that could solve this issue.

Peak Signal-to-Noise Ratio

PSNR is defined in dB as $PSNR = 10LOG_{10}(\frac{MAX^2}{MSE})$, where *MAX* is the maximum pixel value and *MSE* is the Mean Squared Error. This ratio is often used as a quality measurement between the original and a compressed image. In our case, the higher the PSNR, the better the quality of the reconstructed image. This metric is somehow more informative that the straight MSE, but suffers from the same issue. As a result it cannot be our only way of judging our generated images.

Average Log-Likelihood

This metric simply measures the log likelihood that true data can be output by the trained generator. It seems intuitive but reveals unable to quantify image quality (Theis et al. (2016)). On top of that, curse of dimensionality makes density estimation approximation imprecise in higher dimensions. As a result, this measure, which relies on probabilistic density estimation, is more hazardous.

Two measures can be considered as interesting advances in the area of GAN evaluation: Inception Score and Frechet Inception Distance(Lucic et al. (2018)).

Inception Score

The inception score uses a network, called Inception network. It was pretrained on ImageNet to compute logits of generated images (binary predictions). The inception network is a classifier that tells that a given generated image x belongs to class y. With these notation the score is:

$$IS(G) = exp\left(\mathbb{E}_{x\sim p_g}[D_{KL}(p(y|x)\|p(y))]\right) \tag{2.9}$$

where the DKL (Kullback-Leibler Divergence) evaluates how different images from different classes are compared to images from the same class. The inception score does not consider at all the real distribution. As a result, even if it can ensure that images are realistic and diversified enough (when they belong to different categories), they cannot measure how well the generator approximates real images.

Fréchet Inception Distance

The Fréchet Inception distance, proposed in Lucic et al. (2018), also relies on the inception network but tries to solve the absence of real data issue. Instead of using the class label of the generated images, it compares the inception embedding of real images and generated ones. Both are modelled as multivariate Gaussian variables parameterized by mean and covariance. If we denote by (m_r, C_r) and (m_g, C_g) those parameters couples, the Fréchet Inception distance is written:

$$d^2((m_r, C_r), (m_g, C_g)) = \|m_r - m_g\|^2 + Tr(C_r + C_g - 2(C_r C_g)^{1/2}) \tag{2.10}$$

Fréchet Inception distance has the benefit of considering the real distributions. Though, tests (Shmelkov et al. (2018)) have shown that it was inversely correlated with Inception score and suffered from similar issues. One of the main issues with this score (just like the inception score) is that is not possible to relate it to the main characteristics of the generated dataset: for example how realistic or how diverse the images are.

The problem when using Inception score and Frechet Inception distance in the context of brain glioma growth prediction is that the inception network was trained on real life datasets and not at all on MRI of brain gliomas. We can expect that being ”realistic” or ”fit the real data distribution” means a completely different thing for real life pictures and brain MRI, which makes these metrics worthless in our case. It is probably possible though, to train an inception network on brain MRI dataset like BRATS. However, finding a dataset that features enough diversity in terms of brain glioma is not easy.

Precision and Recall

These measures were introduced in Lucic et al. (2018). They rely on the construction of a synthetic data manifold. Then it is possible to compute the distance of an

image to the manifold (typically by finding the distance to the closest point). The definition of the precision in that context is: proportion of generated samples whose distance from the manifold is below a given threshold.

The definition of the recall in that context is: First the latent representations of the generated samples are computed (using the inverted generator). Then recall is the fraction of test samples whose L2-distance to $G(z)$ is below the threshold.

So high recall means that generated images are varied enough to capture most of the manifold while high precision means that generated images are close to real ones. These measures are really intuitive but for real data the manifold is unknown so it becomes impractical (Shmelkov et al. (2018)).

Slice Wasserstien Distance (SWD)

SWD creates randomized projections of the real and generated images and computes the Wasserstein-1 distance between them. It is considered in Shmelkov et al. (2018) to be very useful for comparing high resolution images. But in a context where different GAN architectures may lead to generators that produce images of varied resolution, the SWD is not as relevant.

GAN-train and GAN-test

In Shmelkov et al. (2018), a new method for evaluating GANs is proposed, under the name of GAN-train and GAN-test. GAN-train is the accuracy of a classifier trained on a fake set and tested on a validation set of real images. GAN-test is the accuracy of a classifier trained on the original training set, but tested on the fake set.

The classifier should be trained on labelled images. One thing that could work in the context of this project would be to label MRI images with the type of brain glioma they feature. Such classification requires a good understanding of the images and would lead to meaningful accuracy for GAN-train and GAN-test. This evaluation metric looks very promising for us but involves some labelling (semi-supervised learning).
Ideally GAN-trained accuracy should be close to validation accuracy as it means that GAN images are of similar (so very good) quality as the real images, and also that they are as diverse. GAN-test captures precision: its score being close to validation accuracy means that generated images are close to real data manifold.

Intermediate conclusion on the choice of metric

Comparing the generated images seems like the most natural thing to try. So the reconstruction loss will be applied first. A blurry but sensible prediction seems better than a visually realistic but irrelevant prediction. Depending on how it works we

can try to compare the edge maps of generated and target images in the exact same way (MAE and MSE). Then we can try more sophisticated metrics like GAN-test and GAN-train. As our private dataset is not very large we could label by hand the images (with the type of glioma) in order to train the classifier. It seems also feasible to come up with a custom metric, very specific to this problem, as these get tested.

Chapter 3

Starting Point and Project Direction

3.1 Introduction: Andreas' Work

Andreas was able to apply the Pix2Pix GAN as well as Cycle GAN architecture to MRI slices of brains, in order to predict the evolution of tumors in 2 dimensions. He based his work on the BRATS dataset, which is publicly available, as well as a private dataset given by Dr Matt Williams and prepared by Kammy Pike. Andreas Zinonos managed with Elsa Angelini the registration of MRI images and tested several preprocessing methods to be able to work with the best possible slices. On top of the Pix2Pix he added a time label and used a third network which was trained to predict the time between two MRI brain glioma images. His Pix2Pix GAN with another channel to use this piece of information revealed better than the basic Pix2Pix. Many peprocessing functions that I used to prepare my 3D input were written or heavily inspired by Andreas.

3.2 Dataset

The dataset itself is composed of 3D nii MRI files. It includes images from 36 patients. MRI were acquired for each patient at 2 to 6 different dates. There are T1, T2 and T2-FLAIR images. They are labelled as pre-op, post-op, completed CRT and follow-up(there are a few follow-up dates in most cases). Few MRI cases (around 25) have been segmented by hand and feature a mask indicating precisely where the tumour is. In total there are 330 MRI files of different modalities spread over 148 time points. One advantage that our private dataset has over the datasets that were used in previous challenges like BRATS is that we have sequences of images over time. For each patient, the follow up information should be used to output the best prediction possible.

3.3 Project Direction

The main challenge I fixed for myself on this project was to train a GAN on a 3D dataset. I knew that due to the low amount of data we had, giving up on the multi-

ple slices for each image could cost a lot in terms of prediction quality, not to mention that the output space for my generator would become way bigger, with longer training times etc. However the different slices of a brain are linked together and it did make sense to me to predict them at once. Moreover a 3D predictor would be a way more useful tool for a doctor

The architectures that got my attention as a natural extension for this work are the gEA and dEA GAN architectures. They were suggested by Elsa and are mentioned in Yu et al. (2019) as able to improve the quality of brain glioma MRI images generated by Pix2pix. What they do precisely compared to standard Pix2Pix or even a conditional GAN is explained in the background section.

3.3.1 Preprocessing

After getting used to using efficiently the remote CPU/GPU, the first step was to get the 3D MRI dataset clean and ready to use.
To avoid working on an incomplete and not properly coregistered dataset I reran all the preprocessing functions on the complete raw dataset.

3.3.2 Edge aware GAN models

The EA-GANs github repository that was used (**https://github.com/by-lab/Ea-GANs**) is based on the very well documented Pix2pix Github repository **https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix**, but does not follow the exact same structure and is not documented. I came back to the EA-GAN paper as well as the Pix2pix github repository to understand what I had to set up (visdom and the webpage to display the images for example) and which rescaling needed to be performed on the input images.

The first results were obtained when applying the generator/discriminator Edge Aware GANs scripts on the 3D dataset I had preprocessed.
Then the test images were loaded and evaluated with the most basic metrics.

3.3.3 Edge aware GAN with Time Predictor

The issue that Elsa and I had on that implementation was that the prediction only took into account one input image. Indeed this model takes a pair of real images as its input, the first one acting as the condition of the GAN while the last one is a target. But the time between the date at which the first MRI was done and the date of the second one can be very different in the training dataset, from a few weeks to years. So it was unreasonable to expect very precise predictions from that model.
Just adding the time between the two images as another input would have been a possibility. But the dataset is small enough that that no two pairs of images have the same time difference and I feared that the network would over-fit on this specific date to output the exact target.
So I chose to employ the same method as Andreas. I coded a 3D convolutional network to predict the time difference between two MRI of the same brain (Figure 3.1). Then I added a channel to the edge aware GAN in order to add the time as the input. To avoid the over-fitting problem I used the Time Predictor Network to add another loss to the Edge Aware GAN model. This loss compares the real time difference between the input and the target to the time predicted by the network between the real input and the prediction (Figure 3.2).

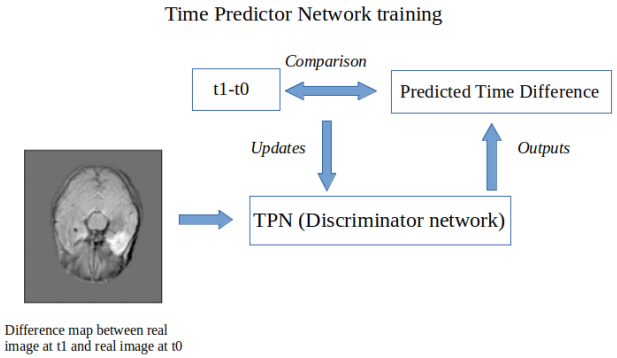


Figure 3.1: Diagram explaining the training of the Time predictor Network

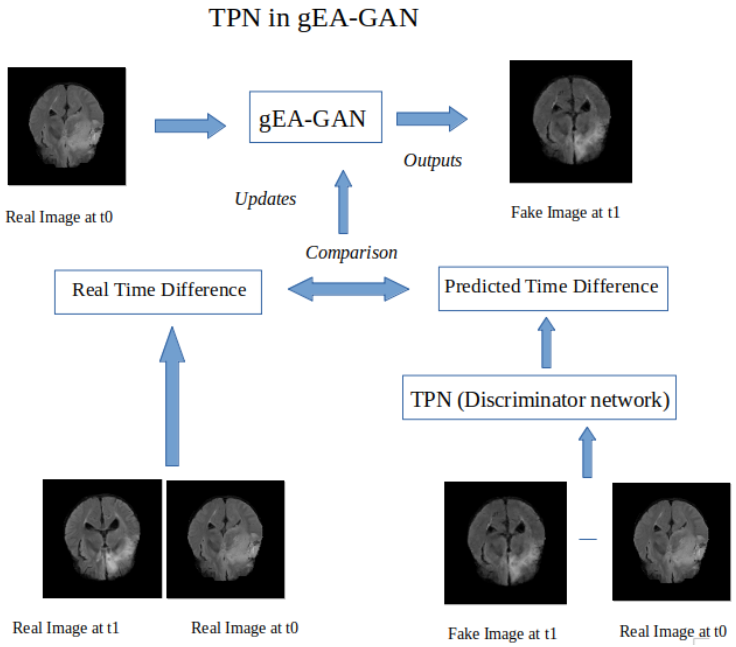


Figure 3.2: Diagram explaining how the TPN loss works

3.3.4 DM-gEA-GAN

The one piece of information that our models had not yet used until that point was that tumour growth was a dynamic evolution. Having and idea of the evolution from instant $t0$ to instant $t1$ is vital to predict the state of the brain at time $t2$. The TPN-gEA-GAN, mentioned above only feeds one image and one time to the network. So the first idea for this new generative adversarial network was to feed it with the image at $t0$, the image at $t1$, and the times $t1 - t0$ as well as $t2 - t1$. In an attempt to reduce the dimension of the input, the natural option was to use one difference map which is $Image\_at\_t1 - Image\_at\_t0$ and the ratio $\frac{t2-t1}{t1-t0}$ for the network to predict the image at $t2$. It seems natural to condense the information like that, for instance if the ratio is 3 that means that given a difference map the situation is likely to evolve 3 times as much before $t2$. We call this model Difference Map gEA-GAN in the rest of the report. It requires a specific step during preprocessing, which is detailed in the Implementation chapter.

Chapter 4

Implementation

4.1 Preprocessing

Preprocessing functions are located in the **Utility** directory. The preprocessing sequence to get the dataset ready for all the projects we used is the following:

1. Brain Masking: **pipeline_mask_FLAIR_patients_v2.sh**
2. Coregistration: **coregister.py**
3. Resampling: **upsampling.py**
4. Pairing: **preprocessing.py**
5. Scaling: **rescaling.py**.

4.1.1 Brain Masking

Brain masking is performed using the shell script **pipeline_mask_FLAIR_patients_v2.sh**.
It uses the atlas **"GG-853-FLAIR-2.0mm"** as well as the mask atlas **"MNI152_T1_2mm_brain_roi"** and relies on **fsutil** to compute the transformation to register the atlas to the given image. Then it applies the same transform to the mask and apply the mask to the initial image. Finally it also applies a very high threshold (the top 0.5%) to eliminate pixels that are abnormally bright. The edge of the masked image when masking is not done perfectly are especially subject to very bright values.

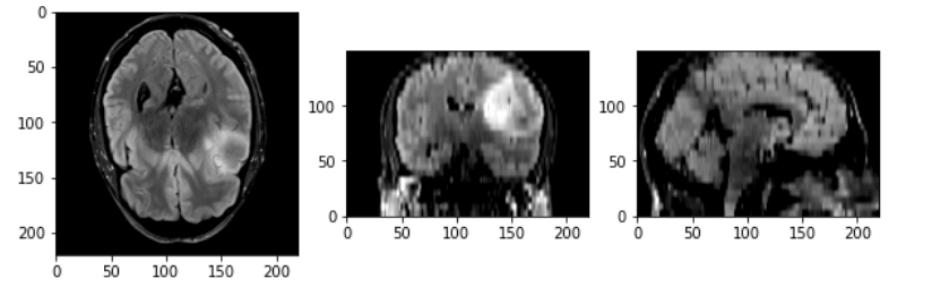


Figure 4.1: Original t2-Flair

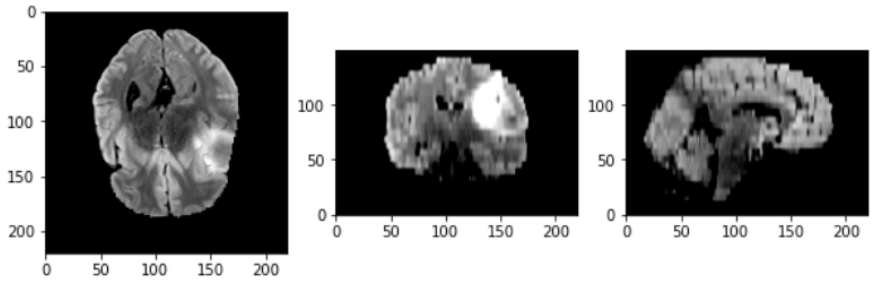
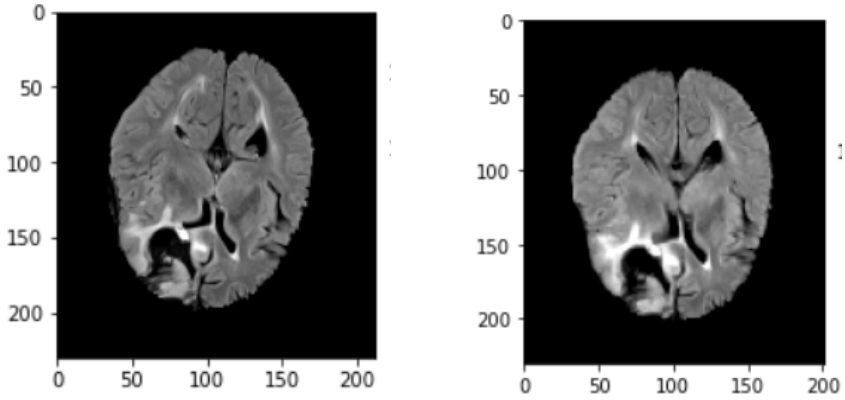


Figure 4.2: Masked t2-Flair

4.1.2 Coregistration

Coregistration is performed in two steps. As the name of the image files include the date, for each patien the first date is found and every other dates are coregistered to that first date. **ImageRegistrationMethod** is a Simple ITK method that does exactly that with some freedom in the parameters like sampling strategy, interpolator, optimizer as gradient descent. The coregistration python file is **coregister.py**.



(a) Before **(b)** After

Figure 4.3: t2-Flair before and after Coregistration

4.1.3 Upsampling

The Edge Aware implementation is primarly designed to work with 128x128x128 images. As the images from the dataset are 256x256x55 there had to be a resampling of some sort. The fonction is called **upsampling.py** and uses the Simple ITK method **sitk.Resample** on every image in the dataset.

4.1.4 Pairing

The Edge Aware models are variations of conditional GANs and take as input pairs of chronologically ordered MRI. In practice we input a 256x128x128 volume with the

first MRI at the bottom and the target on top. So for every patient the dates are ordered and for each eligible pair of dates the 128x128x128 images are merged using the simple ITK method "Paste". The file implementing pairing is **preprocessing.py**. Each pair (Figure 4.4) is named with the number of weeks that separates the two images, so that we can load this time difference with the TPN-GAN data loader. That way the dataset is ready for the TPN-gEA-GAN as well.

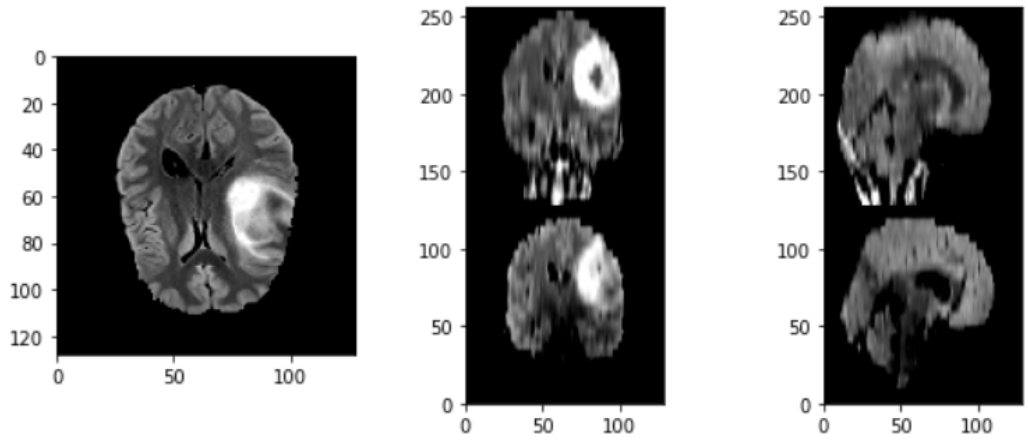


Figure 4.4: Pair of consecutive t2 Flair images from the same patient (the longitudinal slice features the bottom brain)

4.1.5 Split between training set and test set

The split was performed with a balance of 70% training and 30% testing. The results presented in the report are obtained with the split being performed on patients and not images. We realised that the network could over fit and imitate some images from the training set during testing if the split was performed on images.

4.1.6 Re-Scaling

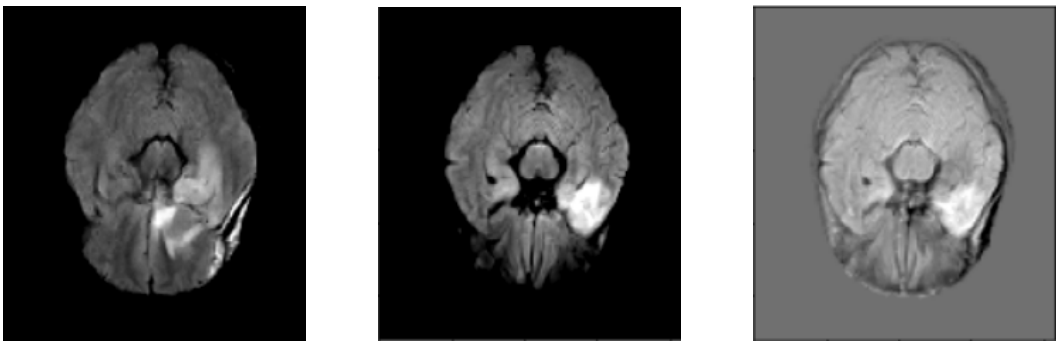
The Edge Aware GANs are able to predict images whose are scaled between -1 and +1. Originally the MRI as they are read by Simple ITK are scaled between 0 and around 1000. I used the python script **rescaling.py** to switch between training/testing ($-1 < x < 1$) and displaying the images ($0 < x < 255$).

4.1.7 Data Augmentation

Some data augmentation methods are employed by default in the Edge Aware GAN repository. They include flip, resize and crop.

4.1.8 For Time Predictor: Difference Map

The time predictor uses a difference map between an image at t_1 and an image at t_2 to predict the time difference. Here is an example of two longitudinal views of MRI from the same patient at two different dates as well as the difference maps between the two. The difference map captures the evolution of the coregistered brain MRI and so is the important piece of information in order to predict the time.



(a) First MRI **(b)** Second MRI **(c)** Difference Map

4.1.9 Alternative pairing for Difference Map gEA-GAN

Only the patients who have at least three t2 scans are considered. Then all the possible ordered triplets (the three MRI must come from the same patient) are extracted. The difference map (image1-image0) is computed and paired to the image at t2. And the pair is labelled with the ratio $\frac{t2-t1}{t1-t0}$.

4.1.10 Alternative Preprocessing with tumour Masking

The dataset comes with a few hand-made tumour masks so a small dataset with 3D tumour only images was preprocessed. The steps were applied in this order:

1. Coregistration together: **mask_coregistering.py**
2. Masking (immediate on command line by using fsl module): **fslmaths 'image_1.nii.gz' -mas 'mask.nii.gz' 'saving_location.nii.gz'**
3. Resampling: **upsampling.py**
4. Pairing: **preprocessing.py**
5. Scaling: **rescaling.py**.

The coregistration step consists in normal coregistration of brain images to the first scan but also coregistration of the tumour masks following the same transformation as the brain images. Then application of the tumour masks. The 3 last steps are the usual preprocessing steps for the Edge Aware GAN mentioned above.

Unfortunately this hand-made masking concerns t1 images only and are available for 7 patients only. Moreover we can see in this dataset some unusual behaviours like tumors getting smaller. It is impossible for the network to predict interesting images in these conditions. And visually the predictions look indeed bad. As a result, focusing on predicting tumour only seemed irrelevant to us and our results are not presented in the report.

4.2 Models and Architectures

4.2.1 Introduction

This chapter describes the different models and architectures used in this project, beginning with the hyper parameters that are shared across the models. The tables detail the architectures of the networks and are accompanied with the corresponding losses. These two pieces of information, combined with the hyperparameters determine precisely the model. The nature of each layer (Convolutions, ReLU, BatchNorm, Linear) are considered known by the reader. Batchnorms, Convolutions and ConTransposes are naturally in 3D.

4.2.2 Hyperparameters

Learning rate

The learning rate lr affects how quickly the networks learns, if it is too high the system can keep oscillating and skip global minima. If it is too low it can take forever to learn, or get stuck in local minima. For my problem lr=0.00002 was relevant. It is the default learning rate in the EA-GANs training.

Beta1

Beta1 affects the momentum of the learning. It is set at 0.5 by default, but due to the small amount of data and low batch size (4 was the maximum not to go over the GPU memory capacity), I increased it to 0.6 to reduce oscillations during training.

Optimizer

Adam is an adaptative learning rate method, that computes individual learning rates for each weight of the neural network using the estimation of first and second moment of the gradient. This estimation relies on moving averages evaluated on a current mini-batch. Adam optimizer has proven very efficient for a number of models and was trusted here without testing other optimizers. It is the default choice in the original EA-GANs implementation.

Sobel lambda

The sobel lambda parameter weights how important the edge maps are in the training loss.This parameter can be set to 0 to train a non edge-aware GAN. I set it to the default value 1.0 each time I wanted to train an edge-aware GAN. Sobel lambda is set to start low but to increase as training goes on.

L1 lambda

L1 lambda parameter weights how important the L1 distance between generated image and target is in the training loss of the generator. By default it was set to 1.0 and seemed to occupy a reasonable proportion in total training loss (magnitude similar to the Sobel loss and about half the generator and discriminator losses). It was kept at 1.0 for the training.

Batch Size

The batch size can be set to 4 maximum with our our machine GPU. I did not experience much difference between a batch size of 1 and 4 in the quality of the images at the end of the training. Obviously though the training loss shows less oscillation when using the bigger batch. We used a batch size of 4. With the redundancy of the images from the training set (all chronological pairs of images are part of the dataset so images can appears several times), this size of batch should not have been much bigger in any case.

Epochs

We achieve decent visual image quality at around 500 epochs. It takes around 10 hours to train. Tests presented in the experimental results section were generated with 1000 epochs training. One training was performed for 2000 epochs but the evaluation metric showed no real evolution.

Dropout

EA-GANs implementation allows us to use or not dropout. Dropout has the following effect: during training some nodes are temporarily ignored. This simulates training several different networks and taking an approximated output between them. Dropout often helps regularizing the network and reduces overfitting, which is especially useful in our case, as our dataset is very small. As a result we used dropout everytime.

Gamma, only for TPN-EA-GANs

The Gamma parameter is the time loss weight for the TPN-gEA-GAN. It weights how important the difference between predicted time and real time is in the training loss of the TPN-GAN generator. Just like Sobel lambda it is set to start low and to increase as training goes on to a maximum of 1.0.

4.2.3 3D Pix2pix

These two tables detail the different layers in the generator and discriminator architectures of our 3D version of Pix2pix.

Generator: UnetSkipConnection

Layer Type	Kernel Size	Stride	Output Channels
Conv	4	2	64
LeakyReLU-Conv-BatchNorm	4	2	128
LeakyReLU-Conv-BatchNorm	4	2	256
LeakyReLU-Conv-BatchNorm	4	2	512
LeakyReLU-Conv-BatchNorm	4	2	512
LeakyReLU-Conv-ReLU-ConvTranspose-BatchNorm	4	2	512
ReLU-ConvTranspose-BatchNorm-Dropout	4	2	512
ReLU-ConvTranspose-BatchNorm-Dropout	4	2	512
ReLU-ConvTranspose-BatchNorm	4	2	256
ReLU-ConvTranspose-BatchNorm	4	2	128
ReLU-ConvTranspose-BatchNorm	4	2	64
ReLU-ConvTranspose-tanh	4	2	1

Discriminator: Convolutional

Layer Type	Kernel Size	Stride	Output Channels
Conv-LeakyReLU	4	2	64
Conv-BatchNorm-LeakyReLU	4	2	128
Conv-BatchNorm-LeakyReLU	4	2	256
Conv-BatchNorm-LeakyReLU	4	2	512
LeakyReLU-Conv-BatchNorm	4	1	512
Conv	4	1	1

LOSS

This part details how the loss is computed.
Prediction takes the value ”True” or ”False” and is the discriminator output

pred_real = prediction(concatenate((real_A, real_B))
pred_fake = prediction(concatenate((real_A, fake_B))

loss_G_GAN = criterionGAN(pred_fake, True)
loss_G_L1 = criterionL1(fake_B, real_B) * lambda_A
loss_G = loss_G_GAN + loss_G_L1

loss_D_fake = criterionGAN(pred_fake, False)
loss_D_real = criterionGAN(pred_real, True)
loss_D = (loss_D_fake + loss_D_real) * 0.5

4.2.4 Generative Edge Aware GAN and Discriminative Edge Aware GAN

The Generative Edge Aware GAN has the same architecture as the basic 3D Pix2Pix here. The discriminator from the Discriminative Edge Aware GAN takes the edge maps as input on top of the rest. Other than that, they are not different. The losses (and so the backpropagations), however, are different. We choose to denote the edge map of the target (real_B) by real_sobel and the edge map of the prediction (fake_B) by fake_sobel.

LOSS: Generative Edge Aware GAN

pred_real = prediction(concatenate(real_A, real_B))
pred_fake = prediction(concatenate(real_A, fake_B))

loss_G_GAN = criterionGAN(pred_fake, True)
loss_G_L1 = criterionL1(fake_B, real_B) * lambda_A
loss_sobelL1 = criterionL1(fake_sobel, real_sobel) * sobelLambda
loss_G = loss_G_GAN + loss_G_L1 + **loss_sobelL1**

loss_D_fake = criterionGAN(pred_fake, False)
loss_D_real = criterionGAN(pred_real, True)
loss_D = (loss_D_fake + loss_D_real) * 0.5

LOSS: Discriminative Edge Aware GAN

pred_real = prediction(concatenate(real_A, real_B,**real_sobel**))
pred_fake = prediction(concatenate(real_A, fake_B, **fake_sobel**))

loss_G_GAN = criterionGAN(pred_fake, True)
loss_G_L1 = criterionL1(fake_B, real_B) * lambda_A
loss_sobelL1 = criterionL1(fake_sobel, real_sobel) * sobelLambda
loss_G = loss_G_GAN + loss_G_L1 + **loss_sobelL1**

loss_D_fake = criterionGAN(pred_fake, False)
loss_D_real = criterionGAN(pred_real, True)
loss_D = (loss_D_fake + loss_D_real) * 0.5

4.2.5 Time Predictor and TPN-Ea-GAN

Time Predictor Network

The time predictor networks take a difference map as input and outputs an expected time difference.

Layer Type	Kernel Size	Stride	Output Channels
Conv-ReLU	4	1	64
MaxPool-BatchNorm	3	3	
Conv-ReLU	4	1	128
MaxPool-BatchNorm	3	3	
Conv-ReLU	4	1	192
MaxPool	2	2	
Flatten-Linear-ReLU-Dropout			50
Linear-ReLU			1

Time Predictor Loss

loss = criterionL2(true_time, prediction)

TPN-Ea-GAN Generator: UnetSkipConnectionTPN

The second part of the U-net has a channel for the time.

Layer Type	Kernel Size	Stride	Out channels	t-input-chan
Conv	4	2	64	
LeakyReLU-Conv-BatchNorm	4	2	128	
LeakyReLU-Conv-BatchNorm	4	2	256	
LeakyReLU-Conv-BatchNorm	4	2	512	
LeakyReLU-Conv-BatchNorm	4	2	512	
LeakyReLU-Conv	4	2	512	
ReLU-ConvTranspose-BatchNorm	4	2	512	
ReLU-ConvTranspose-BatchNorm-Dropout	4	2	512	yes
ReLU-ConvTranspose-BatchNorm-Dropout	4	2	512	yes
ReLU-ConvTranspose-BatchNorm	4	2	256	yes
ReLU-ConvTranspose-BatchNorm	4	2	128	yes
ReLU-ConvTranspose-BatchNorm	4	2	64	yes
ReLU-ConvTranspose-tanh	4	2	1	

The **TPN-Ea-GAN Discriminator** takes as input the time as well, it is the only difference with the Edge Aware GAN described in the last section.

LOSS: Generative Edge Aware TPN-GAN

pred_real = prediction(concatenate(real_A, real_B, true_time))
pred_fake = prediction(concatenate(real_A, fake_B, true_time))
pred_time = TPN_prediction(real_A, fake_B)

loss_G_GAN = criterionGAN(pred_fake, True)
loss_G_L1 = criterionL1(fake_B, real_B) * lambda_A
loss_sobelL1 = criterionL1(fake_sobel, real_sobel) * sobelLambda
loss_G_TPN = criterionL1(true_time, pred_time)
loss_G = loss_G_GAN + loss_G_L1 + loss_sobelL1 + **loss_G_TPN**

loss_D_fake = criterionGAN(pred_fake, False)
loss_D_real = criterionGAN(pred_real, True)
loss_D = (loss_D_fake + loss_D_real) * 0.5

LOSS: Discriminative Edge Aware TPN-GAN

pred_real = prediction(concatenate(real_A, real_B,fake_sobel, true_time))
pred_fake = prediction(concatenate(real_A, fake_B, fake_sobel, true_time))
pred_time = TPN_prediction(real_A, fake_B)

loss_G_GAN = criterionGAN(pred_fake, True)
loss_G_L1 = criterionL1(fake_B, real_B) * lambda_A
loss_sobelL1 = criterionL1(fake_sobel, real_sobel) * sobelLambda
loss_G_TPN = criterionL1(true_time, pred_time)
loss_G = loss_G_GAN + loss_G_L1 + loss_sobelL1+ **loss_G_TPN**

loss_D_fake = criterionGAN(pred_fake, False)
loss_D_real = criterionGAN(pred_real, True)
loss_D = (loss_D_fake + loss_D_real) * 0.5

4.2.6 DM-gEA-GAN

The difference map gEA-GAN uses the network structure of the TPN-gEA-GAN to accommodate the temporal ratio value but keeps the standard gEA-GAN losses as it uses no time predictor.

Chapter 5

Experiments

5.1 3D pix2pix

This model is obtained using the gEA-GAN implementation without Sobel loss and is equivalent to a 3D pix2pix. It is the simplest model we present the results of.

5.1.1 Parameters

- Batch Size: 4
- Model: gEA-GAN
- Flip
- Resize and Crop
- netD: basic conv
- netG: unet 128
- Use Dropout: True
- nb epochs: 1000
- dataset: prepro_128
- lr: 0.00002
- Beta1: 0.6
- Sobel lambda: 0
- L1 lambda: 1.0

5.1.2 Losses

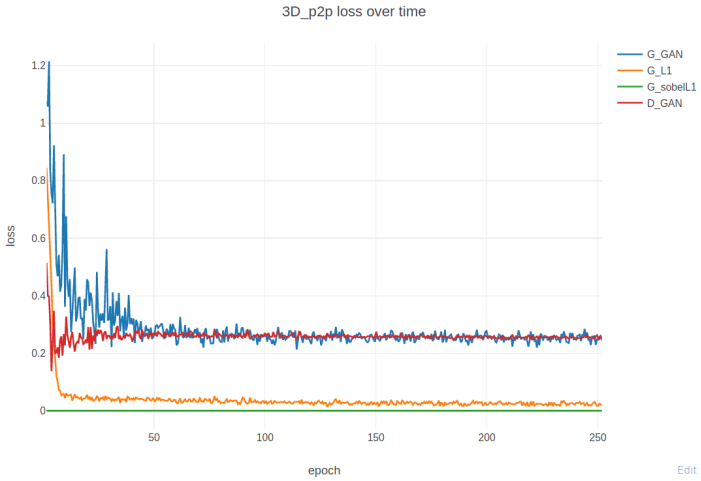


Figure 5.1: 3D pix2pix training loss over epochs

5.1.3 Brain Images from the test set

Across the different models that we trained, we kept the two same training examples (each consisting of a 3D input and target), and presented the longitudinal, coronal and sagittal slices that displayed the tumour the most clearly. The line ”prediction” is actually the fake 3D image generated by the network.
In Figure 5.2 we can see that pix2pix outputs an image of mediocre visual quality, especially the longitudinal slice, but ventricles have approximately the right size and position, and the tumour (bright zone) is not far from the target tumour.

Example 1

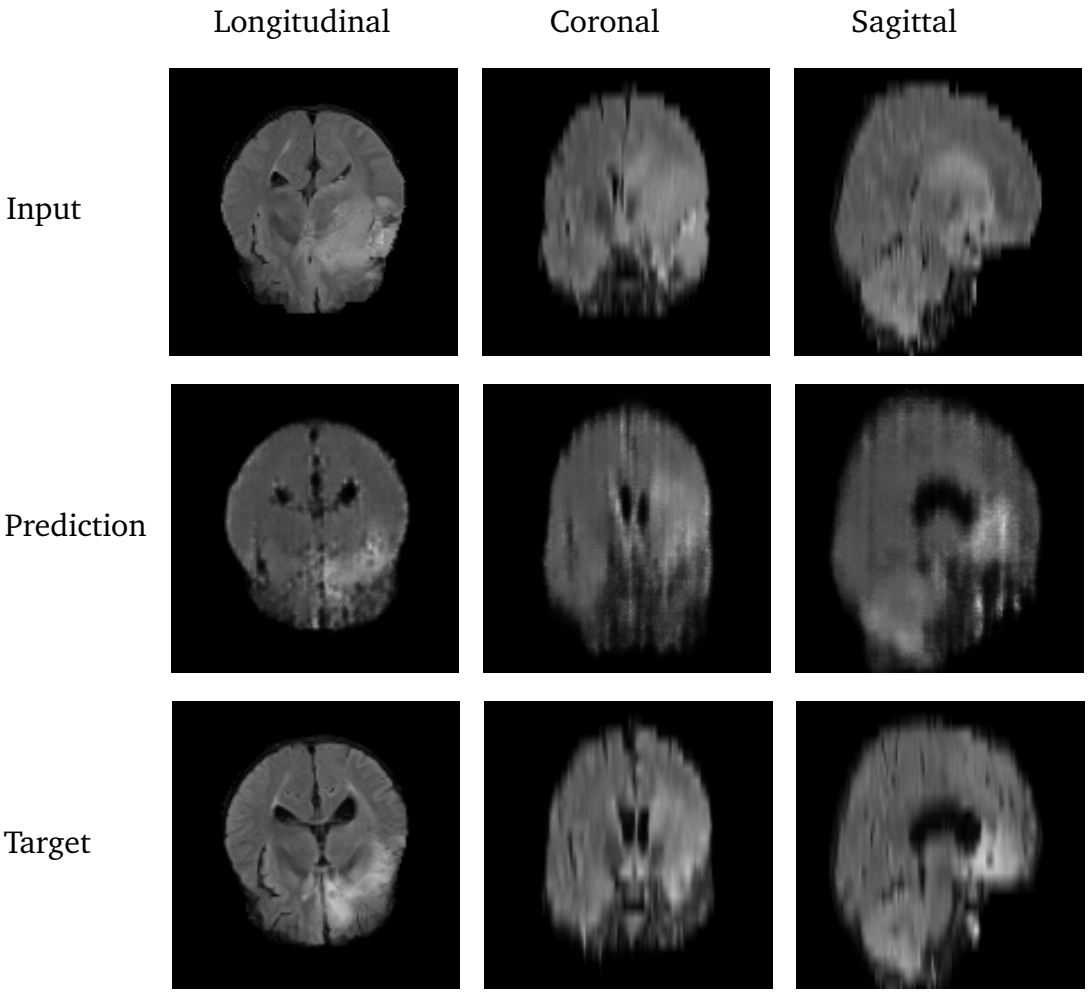


Figure 5.2: pix2pix prediction for example 1

Example 2

The same conclusion seems to apply here, the brain reconstruction is coarse, but the tumour is overall well located. The cavity was not predicted, but it is quite unreasonable to ask with only one image as input.

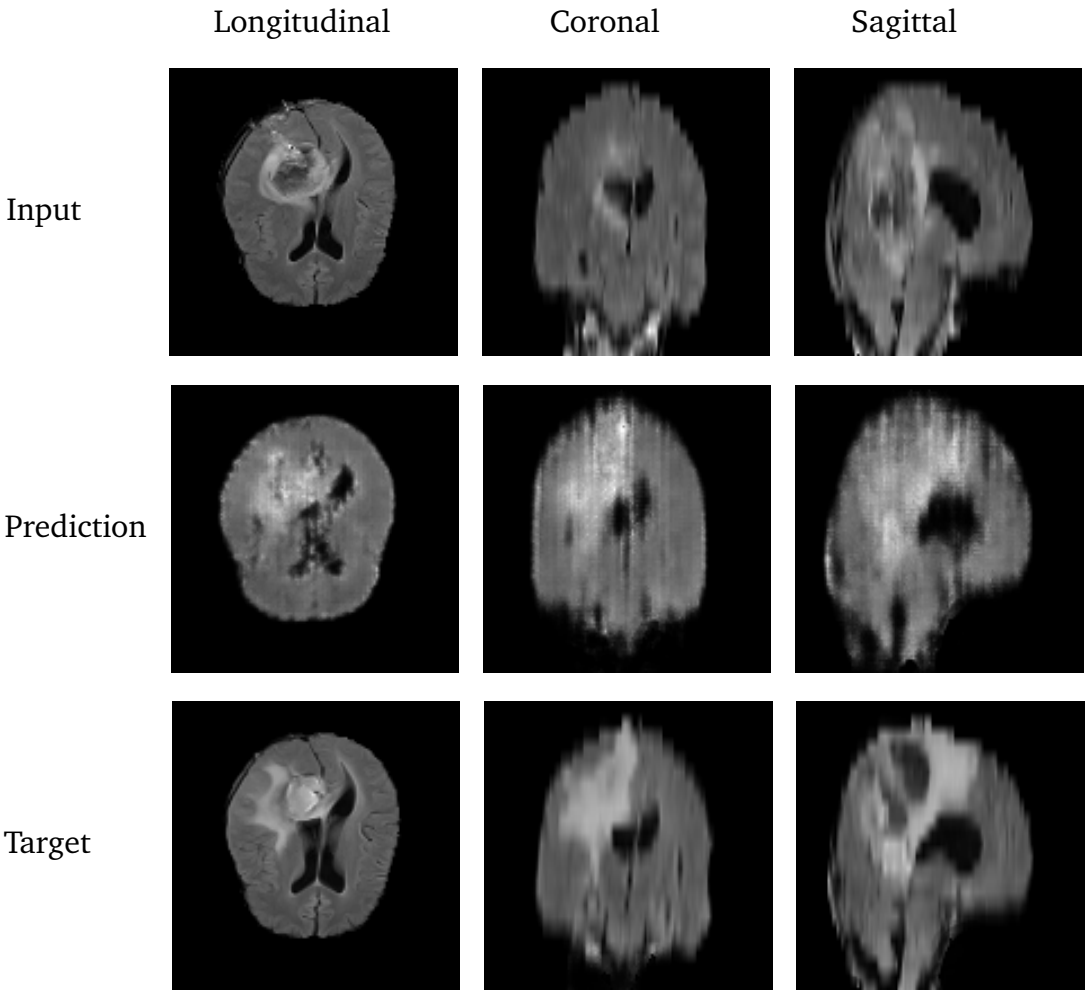


Figure 5.3: pix2pix prediction for example 2

5.2 64-gEA-GAN

This model is the gEA-GAN adapted to a 64x64x64 dataset. It served as an easy-to-train model to experiment with, as well as a potential first step for a multi-scale model(see discussion chapter).

5.2.1 Parameters

- Batch Size: 4
- Model: gEA-GAN
- Flip: True
- Resize and Crop: True
- netD: basic conv
- netG: unet 64
- Use Dropout: True
- nb epochs: 1000
- dataset: preprocessed 64
- learning rate: 0.00002
- Beta1: 0.6
- Sobel lambda: 1.0
- L1 lambda: 1.0

5.2.2 Losses

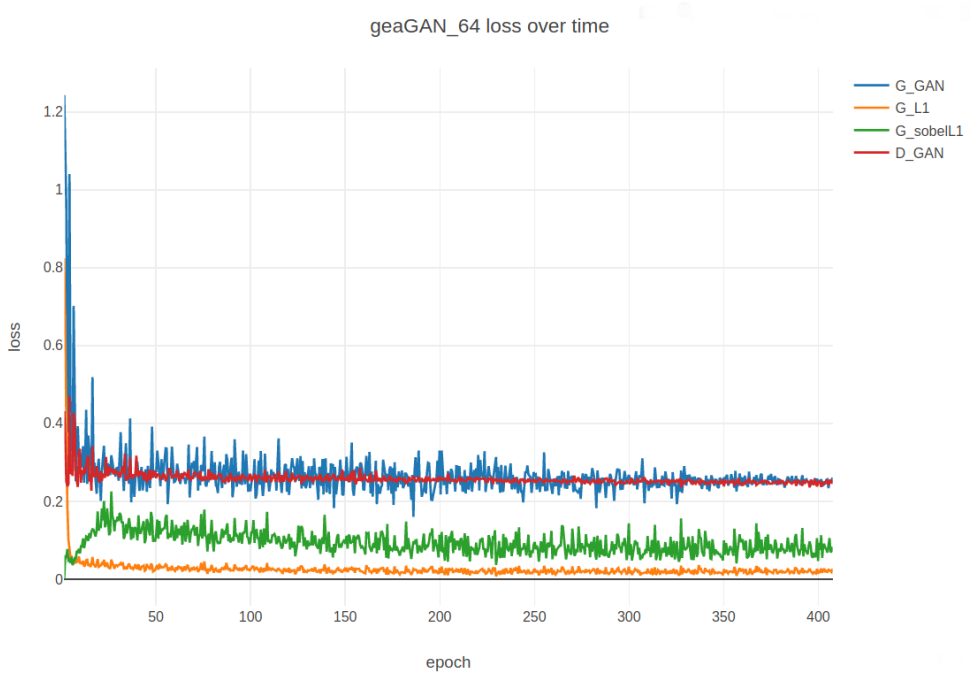


Figure 5.4: 64-gEA-GAN training loss over epochs

5.2.3 Brain Images from the test set

In the Figure 5.5 and 5.6 presented below we can see that even the real image when compressed to 64x64x64 loses all its visual quality (brain convolutions are almost not visible typically). In that regard 64x64x64 seems insufficient. The prediction, given that size, is good though, even the cavity is predicted lightly in example 2.

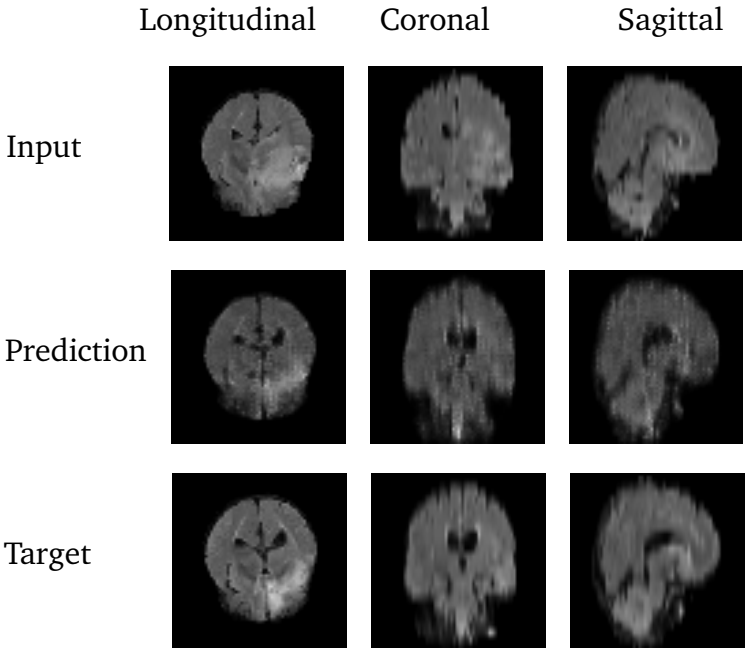


Figure 5.5: 64-gEA-GAN prediction for example 1

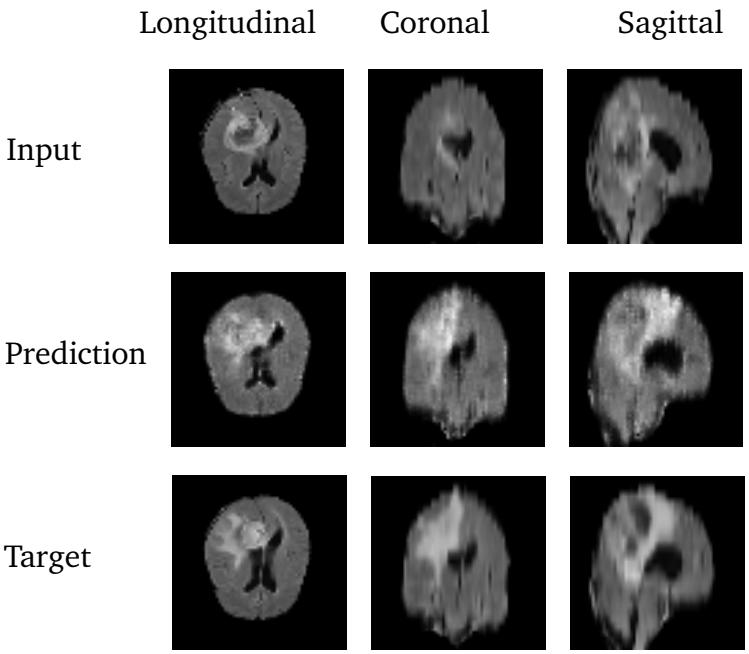


Figure 5.6: 64-gEA-GAN prediction for example 2

5.3 gEA-GAN

The generator Edge Aware GAN is the central model of this report. It is trained to predict 128x128x128 images, and is similar to pix2pix, with a Sobel loss on top for the Generator.

5.3.1 Parameters

- Batch Size: 4
- Model: gEA-GAN
- Flip: True
- Resize and Crop: True
- netD: basic conv
- netG: unet 128
- Use Dropout: True
- nb epochs: 1000
- dataset: preprocessed 128
- learning rate: 0.00002
- Beta1: 0.6
- Sobel lambda: 1.0
- L1 lambda: 1.0

5.3.2 Losses

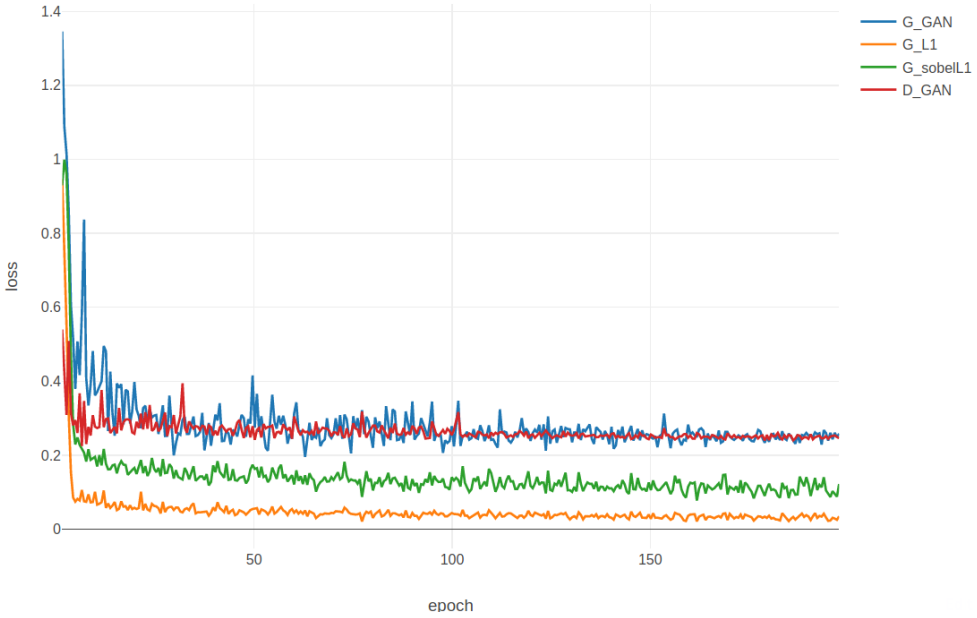


Figure 5.7: gEA-GAN training loss over epochs

5.3.3 Brain Images from the test set

Example 1

On Figure 5.8 one can see that the prediction is of good visual quality. The ventricles are well shaped, even the convolutions are visible. The look of the predicted tumour is a little dark, and its shape is not perfect. But it is still good, especially given the fact that the target is relatively far from the input.

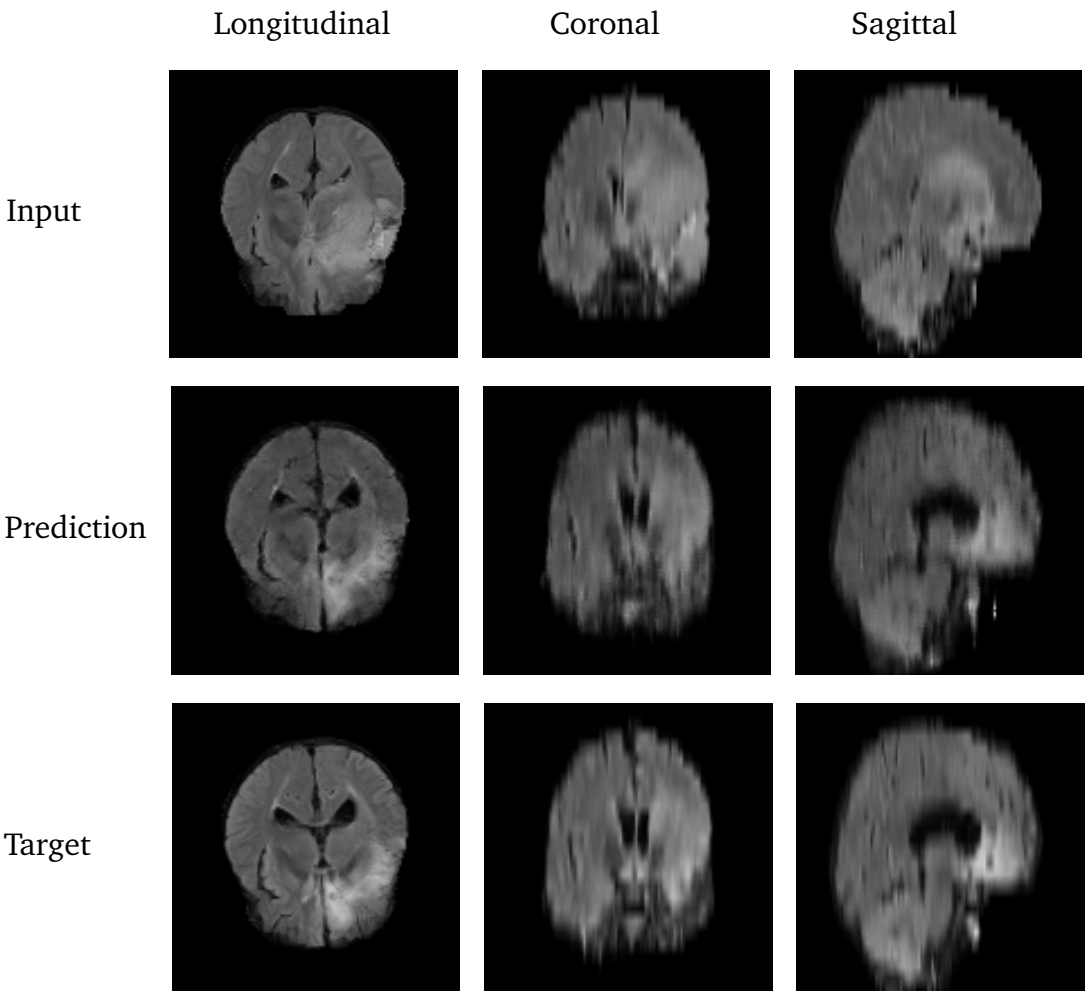


Figure 5.8: gEA-GAN prediction for example 1

Example 2

In Figure 5.9, example 2 confirms a good overall image quality and brain representation, as well as tumour evolution that is close to reality.

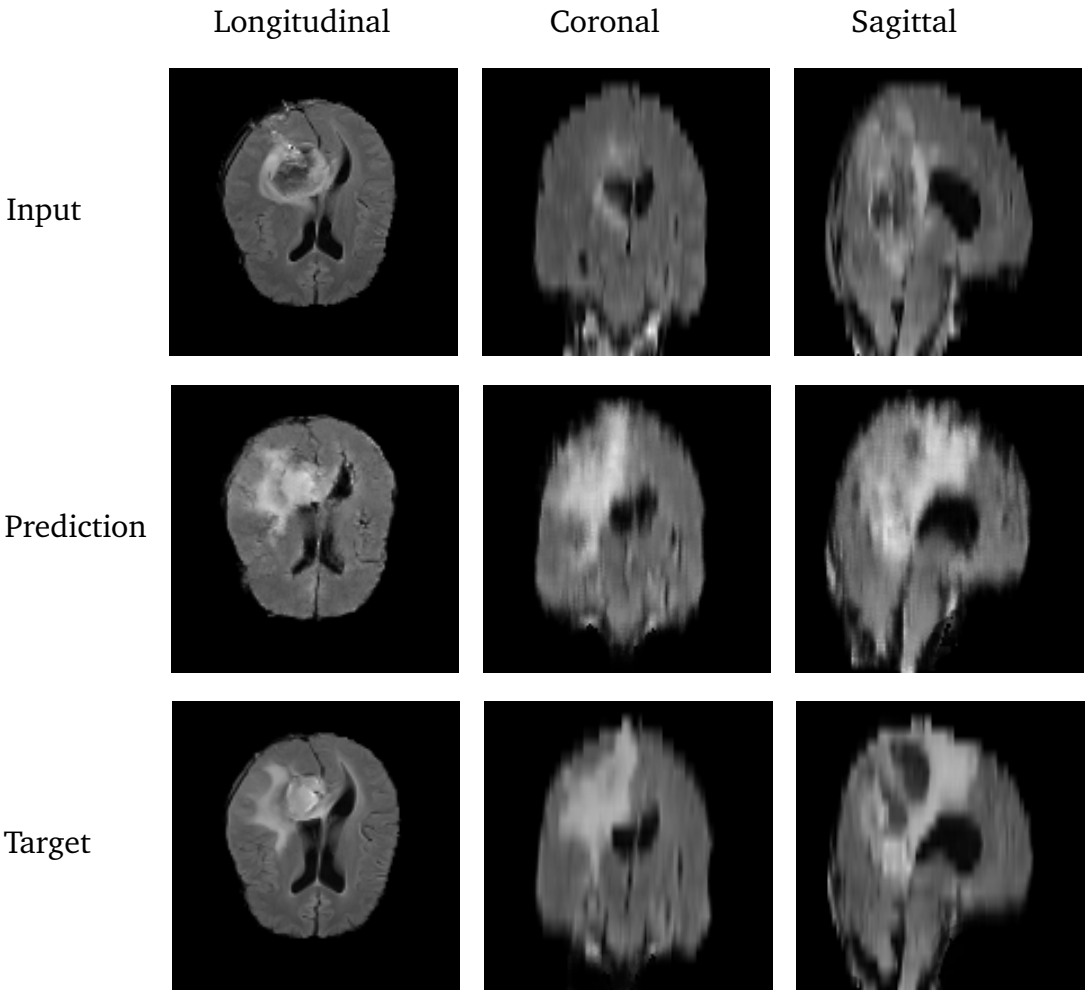


Figure 5.9: gEA-GAN prediction for example 2

5.4 dEA-GAN

The discriminative Edge Aware GAN is a more complex version of the edge Aware GAN as the Sobel maps are also taken into account by the dicriminator.

5.4.1 Parameters

- Batch Size: 4
- Model: dEA-GAN
- Flip: True
- Resize and Crop: True
- netD: basic conv
- netG: unet 128
- Use Dropout: True
- nb epochs: 1000
- dataset: preprocessed 128
- learning rate: 0.00002
- Beta1: 0.6
- Sobel lambda: 1.0
- L1 lambda: 1.0

5.4.2 Losses

The training losses, despite using the same hyper parameters, oscillates way more than for the other models.

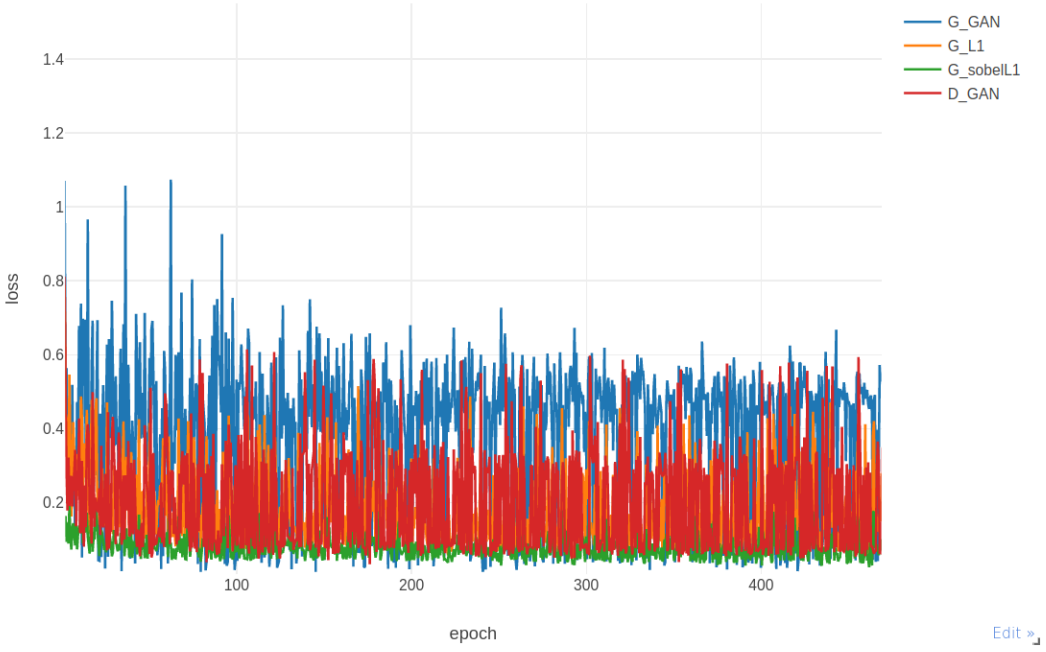


Figure 5.10: dEA-GAN training loss over epochs

5.4.3 Brain Images from the test set

Example 1

Figure 5.11 shows predictions from the dEA-GAN. The images are smooth but also a little more blurry than those formed by the gEA-GAN. Moreover coronal and sagittal views show a strange stripe effect. My guess is that the edges take a too much importance and are avoided in the longitudinal views (hence this blurry look). Moreover the network also probably spots the upsampling that was performed to go from 25 slices to 128 and displays the initial slices. Maybe changing the upsampling approach would help, lowering the Sobel lambda parameter could also work. The shape of the tumour growth prediction is however good.

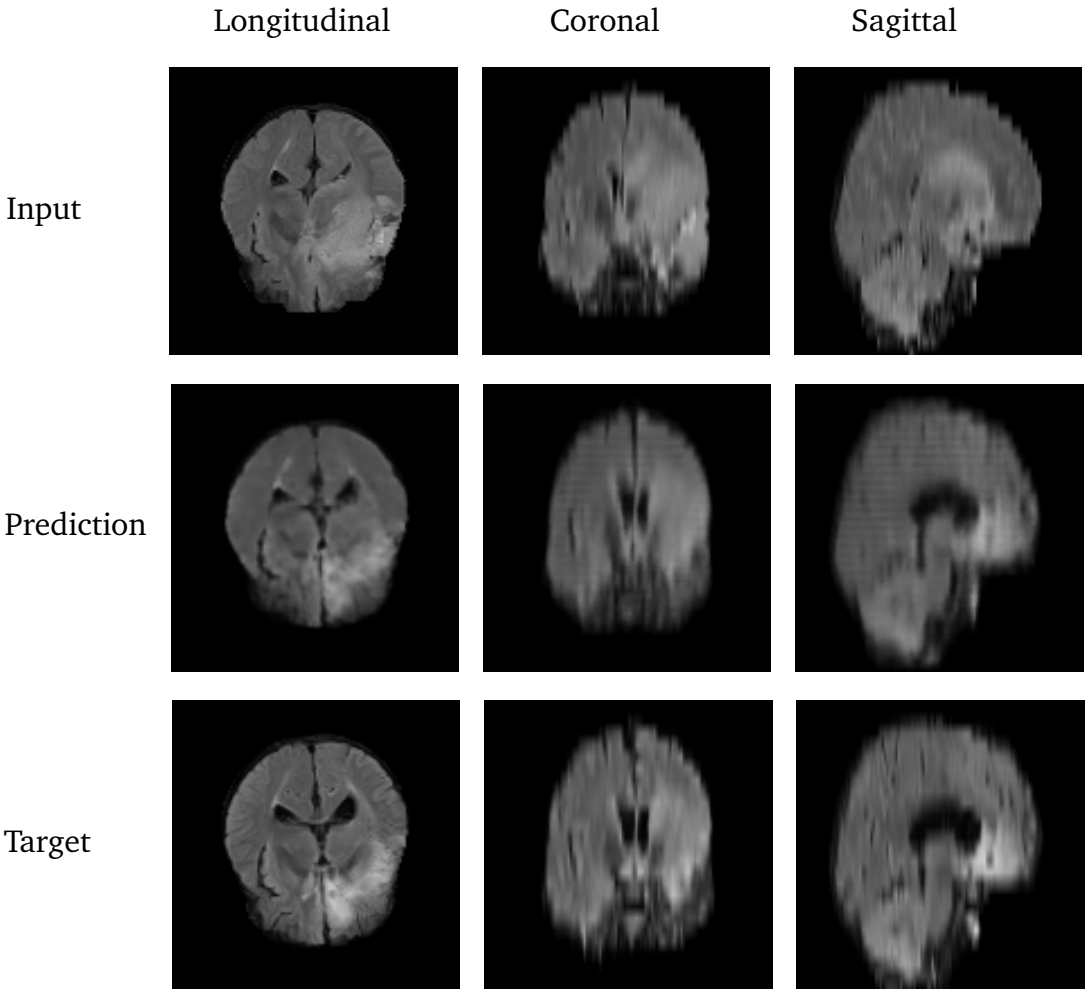


Figure 5.11: dEA-GAN prediction for example 1

Example 2

The same remarks apply here. The tumour growth prediction lacks some contrast but is very well shaped.

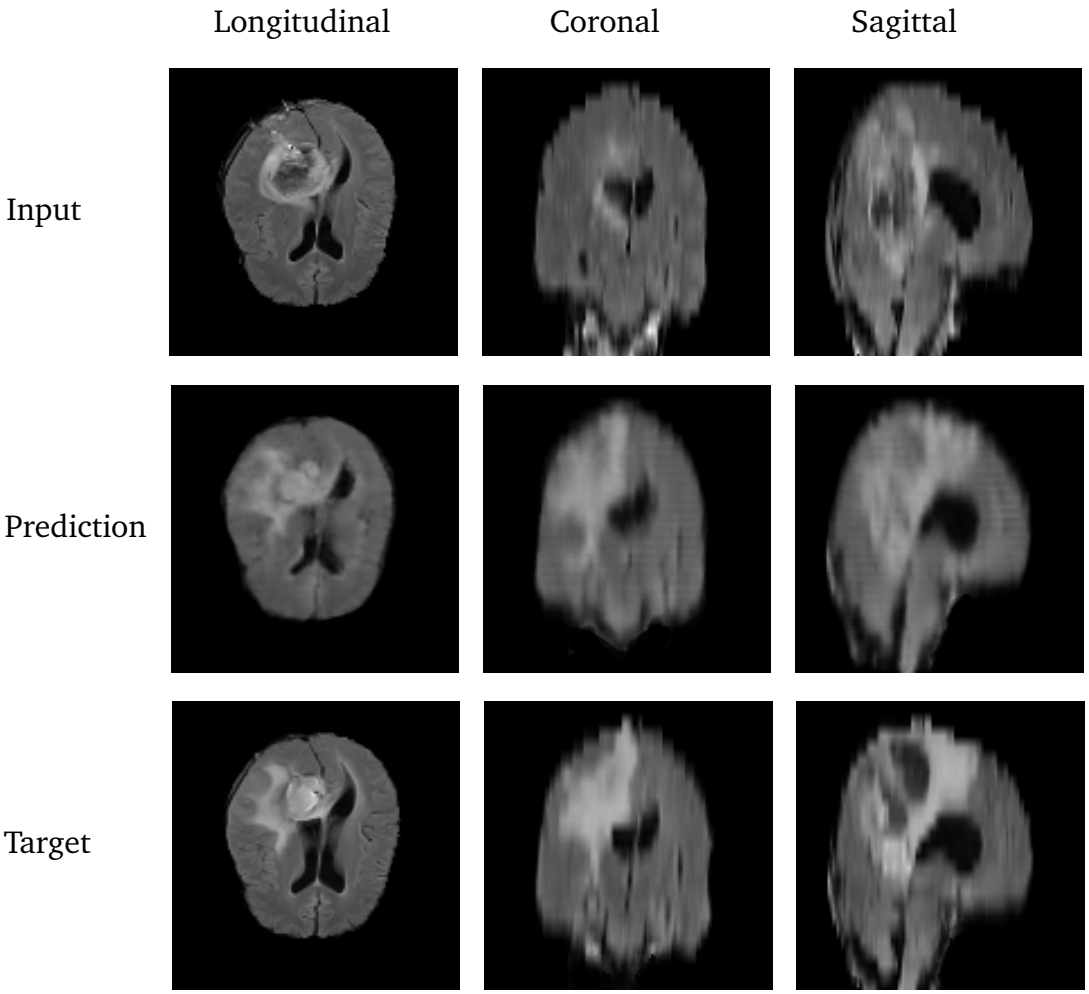


Figure 5.12: dEA-GAN prediction for example 1

5.5 Time predictor

The time predictor is a smaller model that just outputs the expected time between 2 3D brain tumour images. It generates no image.

5.5.1 Parameters

- Batch Size: 4
- Model: Time predictor
- Flip: True
- Resize and Crop: True
- net: Convolutional
- Use Dropout: True
- nb epochs: 200
- dataset: difference map 128
- learning rate: 0.00002
- Beta1: 0.6
- Sobel lambda: 1.0
- L1 lambda: 1.0

5.5.2 Loss

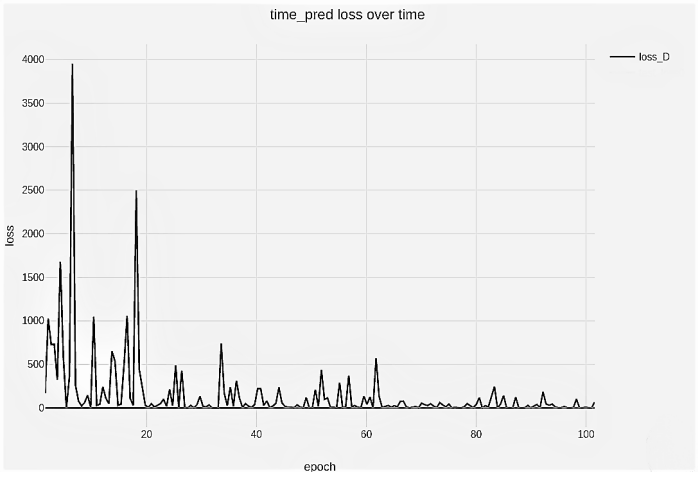


Figure 5.13: Time Predictor training loss over epochs

5.6 TPN-gEA-GAN

The TPN-gEA-GAN uses the trained Time Predictor to take into account the time separating the input and the prediction.

5.6.1 Parameters

- Batch Size: 4
- Model: TPN-gEA-GAN
- Flip: True
- Resize and Crop: True
- netD: basic conv
- netG: unet 128
- Use Dropout: True
- nb epochs: 1000
- dataset: preprocessed 128
- learning rate: 0.00002
- Beta1: 0.6
- Sobel lambda: 1.0
- L1 lambda: 1.0

5.6.2 Losses

Figure 5.14: TPN-gEA-GAN training loss over epochs

The loss reveals that predicting images that look realistic to the discriminator (and close enough to the target) seems to go in the opposite direction from the TPN request. It could be due to a poor time predictor or behaviours and dynamics of brain tumour growth from the dataset that are too diverse to be abstracted and generalised. The TPN loss risks to have brought a lot of noise to the training. We can expect a lower image quality.

5.6.3 Brain Images from the test set

We can see on Figure 5.15 below that indeed the overall quality of the image is slightly lower than when using the simple gEA-GAN. However the tumour is very well positioned and contrast is almost the same as in the target image. The coronal and sagittal view typically show very good tumoral growth prediction.

Example 1

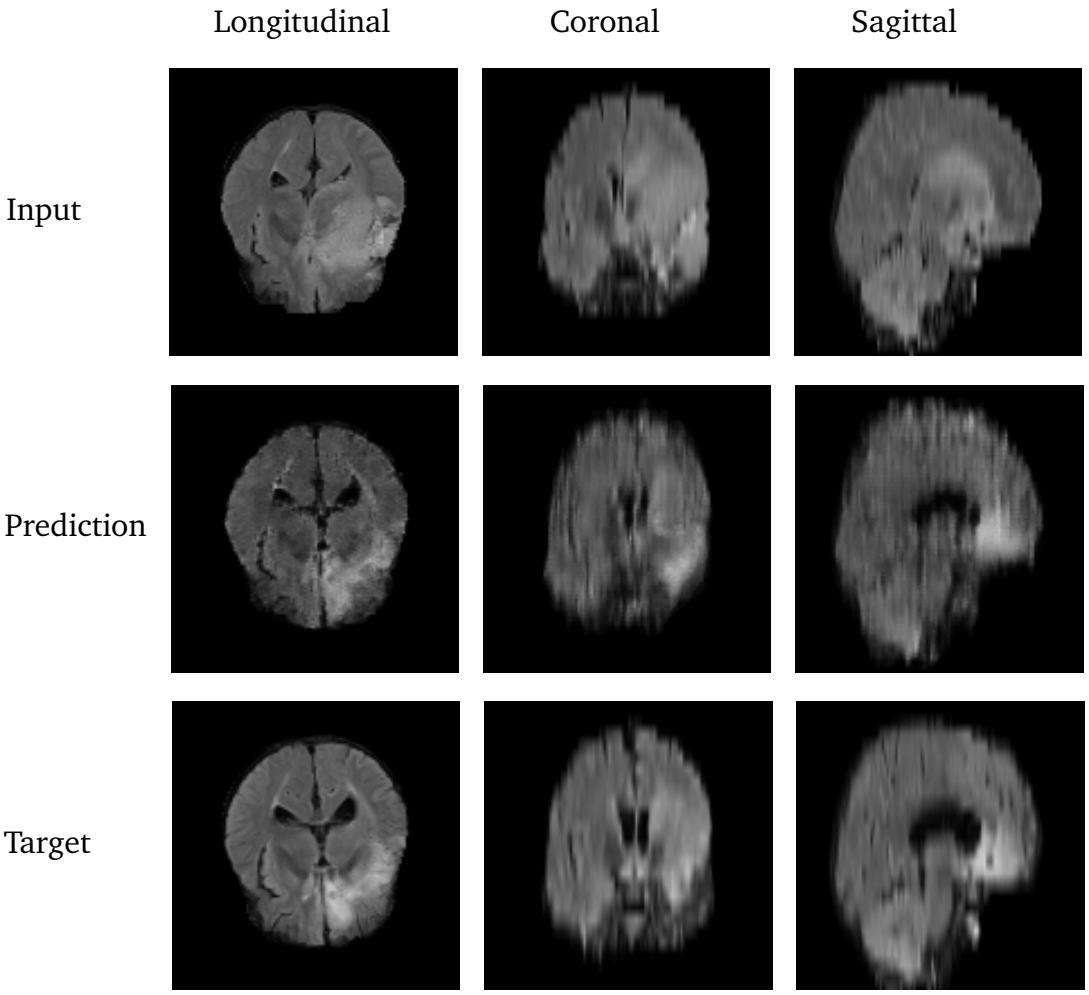


Figure 5.15: TPN-gEA-GAN prediction for example 1

Example 2

In TPN-gEA-GAN, while we can see again thtat the overall visual quality is slightly inferior (the brain convolutions typically do not really appear), the shape of the predicted tumour is rather good for example on the longitudinal view, the tumoral expansion with 4 branches appears clearly on the fake image.

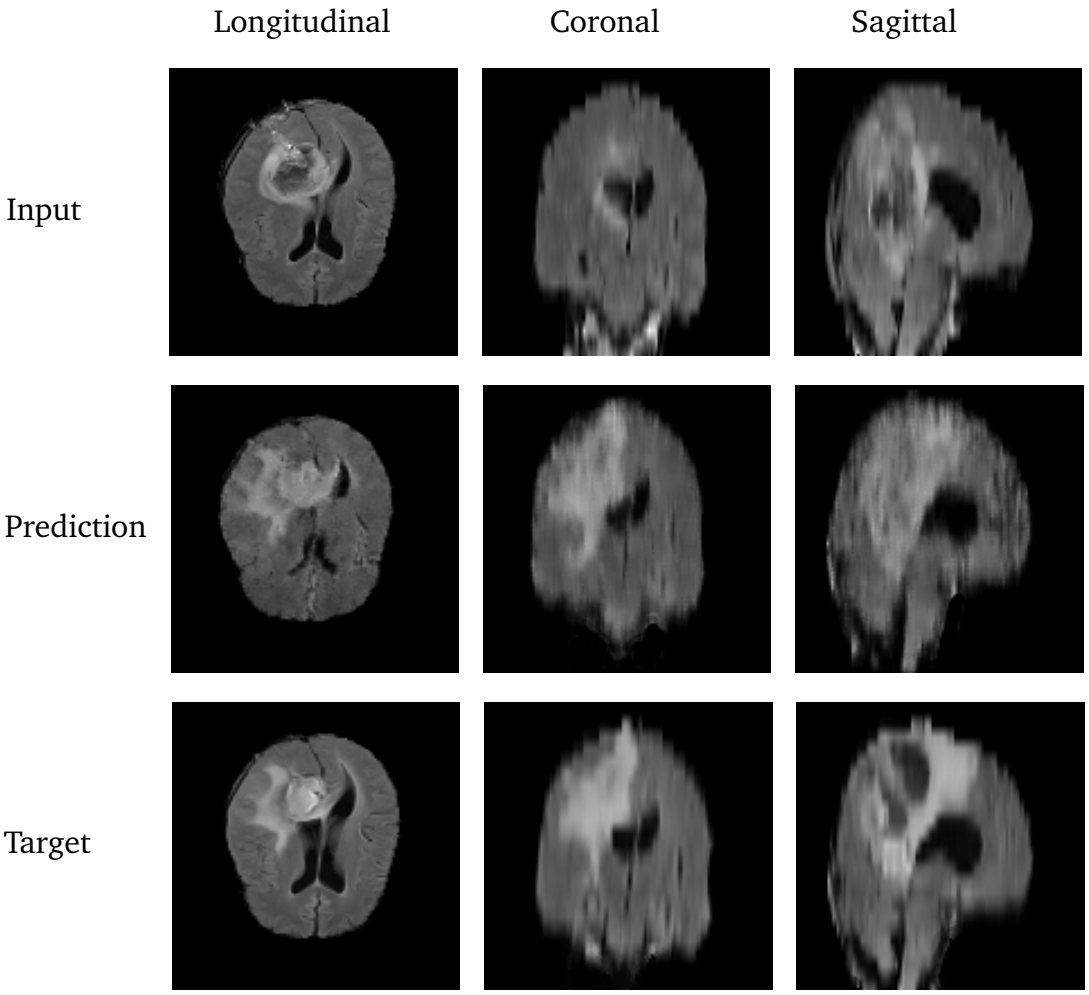


Figure 5.16: TPN-gEA-GAN prediction for example 2

5.7 Visual global comparison

Figure 5.17 and Figure 5.18 gather for example 1 and example 2 the fake images generated by all the models but the DM-gEA-GAN which will be part of the next section.

Overall the visual quality seems to go in that order:

$pix2pix < dEA\_GAN < TPN\_GAN < gEA\_GAN$

while prediction quality (without considering contrast too much but rather the exact shape of the tumour) seems to go in that order:

$pix2pix < dEA\_GAN = gEA\_GAN < TPN\_GAN$

3D Pix2pix has the lowest visual quality and this result was expectable as it is the simplest model of the list, as well as TPN-GAN outputting good tumout growth prediction is coherent with the objective. The dEA-GAN is a little bit disappointing but different hyperparameters could probably be used for better results.

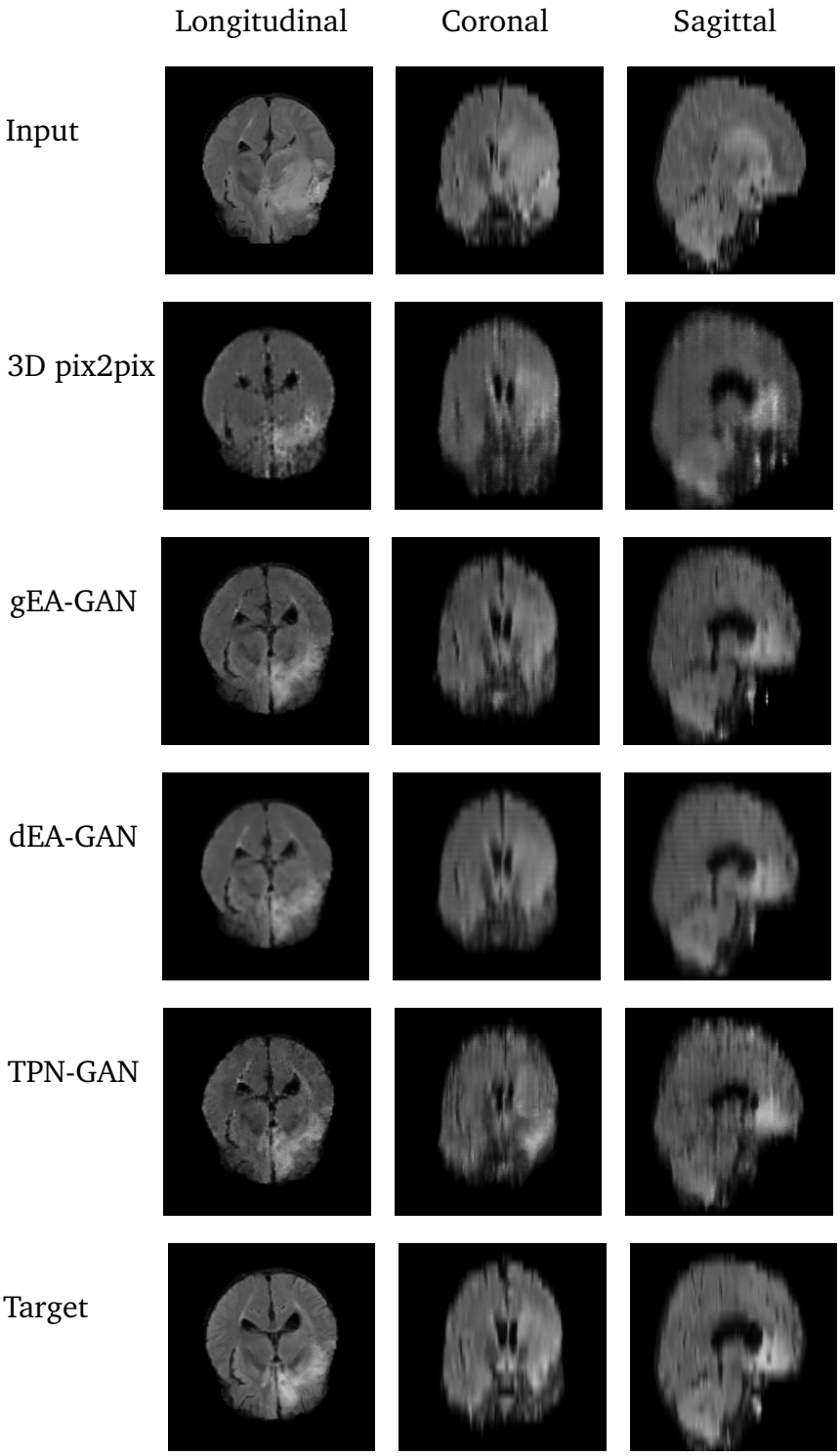


Figure 5.17: Example 1: Comparison between the predictions from the different models

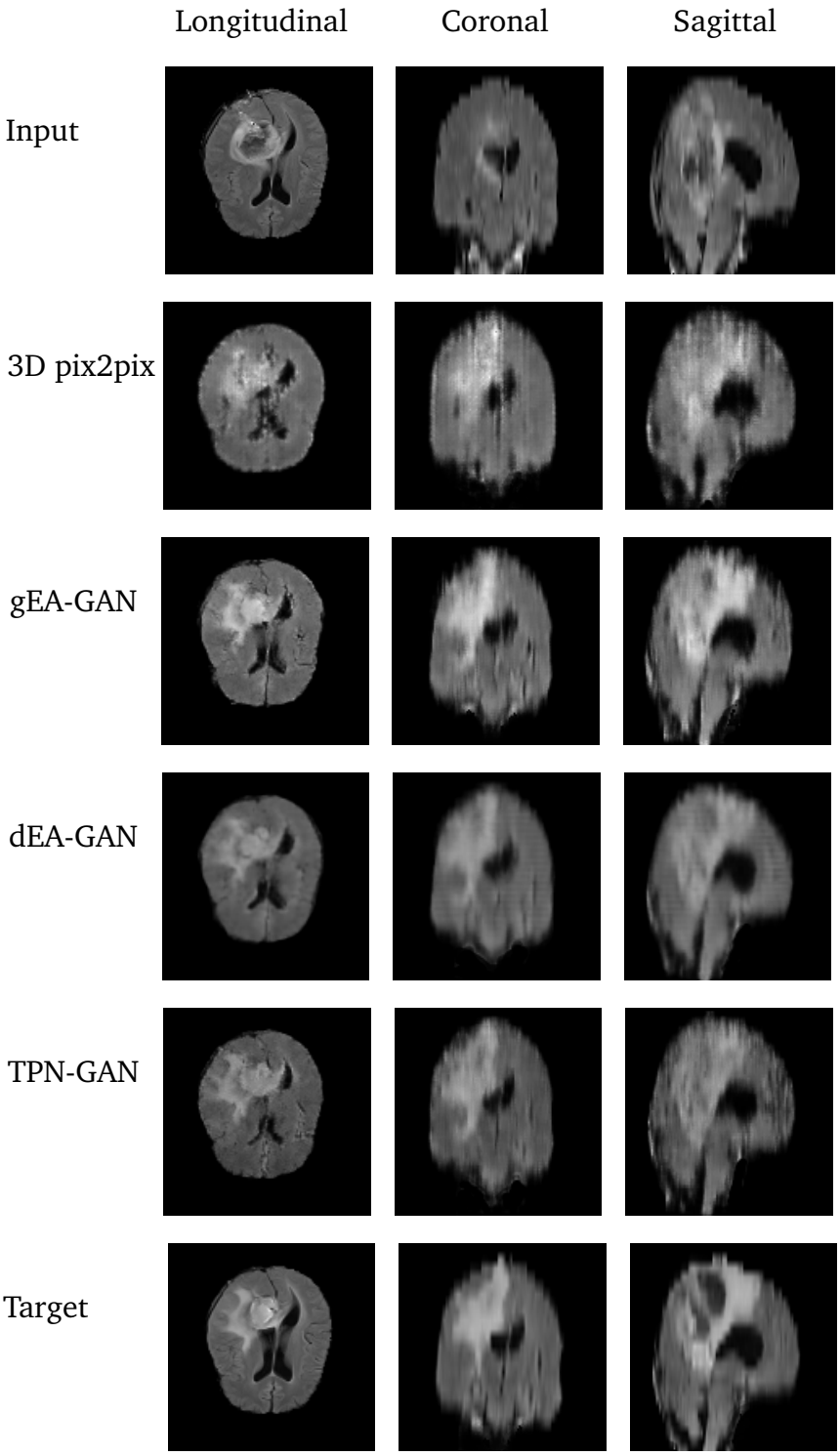


Figure 5.18: Example 2: Comparison between the predictions from the different models

5.8 DM-gEA-GAN

The DM-gEA-GAN is a gEA-GAN model that uses as input a time ratio and difference map. Due to limited time it was only tried with 500 epochs and as such is not to be compared to all the other models.

5.8.1 Parameters

- Batch Size: 4
- Model: diff-gEA-GAN
- Flip: True
- Resize and Crop: True
- netD: basic conv
- netG: unet 128
- Use Dropout: True
- nb epochs: 500
- dataset: preprocessed 128
- learning rate: 0.00002
- Beta1: 0.6
- Sobel lambda: 1.0
- L1 lambda: 1.0

5.8.2 Loss

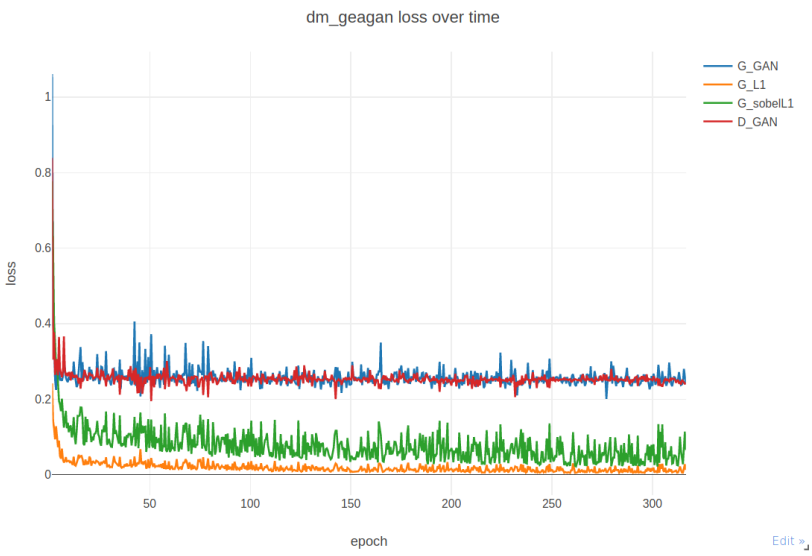


Figure 5.19: Time Predictor training loss over epochs

5.8.3 Visuals

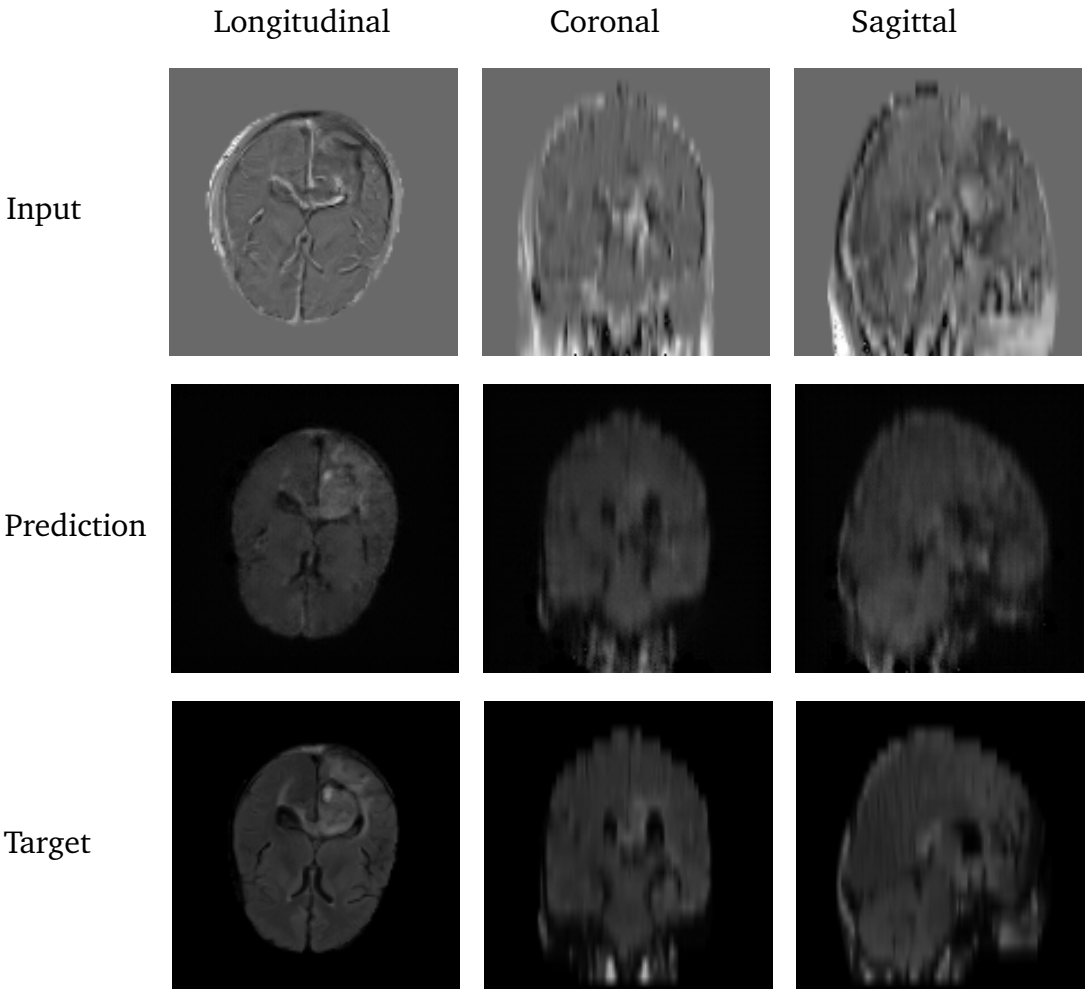


Figure 5.20: DM-gEA-GAN prediction visuals

For a model that has been trained for half as long as all the others, I found the results surprisingly good especially given the fact that the model has no brain input to copy, it just receives an evolution (difference map). I think that these visuals are very encouraging to add different inputs to a gEA-GAN model. For example, my guess is that, if given enough time to train, a model receiving t1 and t2 as well as the time differences to predict t3 would be extremely good.

Chapter 6

Evaluation Metric

INTRODUCTION: on the importance of the metric when evaluating GAN

The specificity of a GAN is that there is no objective loss function used to train it. The way the generator is evaluated is given by the ever changing discriminator. Telling how good a GAN is, is not as obvious as for a model which training is based on a converging loss. While GAN results often look good visually, training a GAN without a good evaluation metric to compare the results at the end has little scientific value.

6.1 Final choice of metrics

We wanted our search for the best evaluation metric to ultimately lead to a loss designed specifically for brain tumors. For instance Kamnitsas et al. (2016) uses contrast and tumour segmentation to design a new evaluation metric.
After going through several ways of evaluation conditional GANS like in Taylor et al. (2019) with the Frechet Inception score, as well as precision and recall, I chose 3 main characteristics that I estimated to be the most important in the generated images. I then derived very simple algorithms to implement them on t2-Flair images.

6.1.1 Expectations

In this project we require 3 main elements to be evaluated by our metric. They are encoded as a 3D vector but can of course be combined to form one general evaluation metric.

- First, the generated brain image should be of good visual quality, that is to say similar to real brain images. A time-testing way of ensuring this is to use the Structural Similarity Index Metric (SSIM) between the generated prediction and the real image target.
  This metric is usually computed to quantify image quality degradation caused by processing such as data compression. I believe it will be very efficient to judge how realistic the brain we are generating are. Peak Signal to Noise Ratio (PSNR) was another option but it is a variation of the MSE and still

concentrates on pixel-by-pixel comparison. SSIM, on the contrary, is correlated with the quality and perception of the human visual system.

- Then, the generated brain image should be close enough to the input image, when focusing on the part of the image which does not correspond to the tumour -or edge of the brain when masking is not perfect- (with intensity above a given threshold) in either the input or the prediction. That area should be more or less the same in those two images. We understand that this may not seem a priority, but due to the fact that we did not have enough segmented brain tumors to predict only the tumors, we predicted full brains. So we also need to judge the brain reconstruction. We should also keep in mind that evaluating this means in a way evaluating the coregistration method: it is likely that not all the target images were perfectly registered to the input images, so the GAN has never tried to achieve perfect reconstruction.
  A metric that would fit this task would in my opinion be a variant of the Distance to Population Score (which is basically a "nearest neighbour" metric), setting a weight of zero to the tumour part. The algorithm is detailed below.

- Finally, the evolution of the tumour is probably the information we are the most interested in. To evaluate how good the prediction is, we will also apply a Distance to Population score between the prediction and the target image, this time giving more weight to areas where at least one image (the target or the prediction) has brighter pixels. Depending on the results when using only prediction image vs target image, we can try computing Distance to Population Score between the difference maps instead (prediction or target minus the input)

6.1.2 Algorithms and Formulas

SSIM

SSIM index is calculated on different image windows. The measure between two images x and y is

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{6.1}$$

where:
μ_x and μ_y are the average intensity values within x and y
σ_x^2 and σ_y^2 are the variance of intensity values within x and y
σ_{xy} is the covariance of intensity values within x and y
C_1 , C_2 are two variables to stabilize the division.

Brain Reconstruction Score

To obtain our brain reconstruction score we will need this weighted distance between a prediction image Y and an input image X.

$$\mathcal{D}_{brs}(X,Y) = \frac{1}{Card(I)} \sum_{i \in I} |X_i - Y_i| \tag{6.2}$$

Where I represents the ensemble of the pixels i such that, given a threshold T, $X_i < T$ **and** $Y_i < T$.
Using this distance we can compute the reconstruction score:

1: The test set is indexed by i going from 1 to N.
2: The ith input image from the test set is denoted by X_i
3: The image generated conditioned on X_i is denoted by Y_i
4: Score=0
5: **for** $i = 1$ to N **do**
6: **if** $argmin_j \mathcal{D}_{brs}(X_i, Y_j) = i$ **then**
7: $Score \leftarrow Score + 1$
8: **end if**
9: **end for**
10: Return $Score/N$

Algorithm 1: Brain Reconstruction Score

Basically, this algorithm returns the proportion of prediction images whose non tumoral areas are the closest (based on MAE distance) to the non tumoral areas from the input image they were predicted from.

tumour Evolution Score

For the tumour Evolution Score we will need this slightly different weighted distance between a prediction image Y and a target image Z.

$$\mathcal{D}_{tes}(Z,Y) = \frac{1}{Card(I)} \sum_{i \in I} |Z_i - Y_i| \tag{6.3}$$

Where I represents the ensemble of the pixels i such that, given a threshold T, $Z_i > T$ **or** $Y_i > T$.
Using this distance we can compute the evolution score:

1: The test set is indexed by i going from 1 to N.
2: The ith target image from the test set is denoted by Z_i
3: The image generated from the ith input and targeting Z_i is denoted by Y_i
4: Score=0
5: **for** $i = 1$ to N **do**
6: **if** $argmin_j \mathcal{D}_{tes}(Z_i, Y_j) = i$ **then**
7: $Score \leftarrow Score + 1$
8: **end if**
9: **end for**
10: Return $Score/N$

Algorithm 2: tumour Evolution Score

If opting for using the difference maps, and denoting by X_i the ith input image in the testing set, just replace X_i and Y_i by respectively $Z_i - X_i$ and $Y_i - X_i$ in the tumour evolution metric algorithm.

6.1.3 Implementation and combination of the metric scores

A python class called **Evaluation** is present in the file **eval.py**. It computes the standard losses MAE, MSE, PSNR, SSIM. MAE, PSNR and MSE were mentioned in the background section and are straightforward to implement. We will use them as a backup metric and be very wary of whether or not they correlate well with our specially designed metric. SSIM is part of our final metric.
Keeping the score 3-dimensional will preserve all the information but if we aim at obtaining a final single score we an imagine two ways of combining these scores.

$$\mathcal{FS} = \lambda_1 SSIM + \lambda_2 BRS + \lambda_3 TMS \tag{6.4}$$

We can choose the lambdas to obtain the balance we want but require:

$$\lambda_1 + \lambda_2 + \lambda_3 = 1 \tag{6.5}$$

Since we want the 3 scores to be high enough, we could prefer this combination:

$$\mathcal{FS} = min(SSIM, BRS, TMS) \tag{6.6}$$

6.2 Experimental Results

6.2.1 All scores comparison

For all the distance involved in the evaluation metrics ”Brain Reconstitution” and ”Tumour Evolution” we used a threshold of 100 (on a 0-255 range). Also, as the DM model takes no MRI as input, the Brain Reconstruction score would not be very valuable.

	MAE	SSIM	Brain Recons	Tumour Evol
Pix2Pix	3.1512108	0.7143752776	0.403975468	0.512820512
64gEA-GAN	3.2045333	0.7446164879	0.538461538	0.487179487
gEA-GAN	3.1396134	0.7264548522	0.461538461	0.543682872
dEA-GAN	2.8334658	0.7418378386	0.461538461	0.512820512
TPN-gEA-GAN	3.741733	0.7085575689	0,435897436	0,564102564
DM-gEA-GAN	3.4622732	0.7167884296	n/a	0.512820512

6.2.2 Total score comparison

The total score chosen is the mean of the SSIM, Brain Reconstruction and Tumour Evolution scores.

	Total Score
Pix2Pix	0.5437
64gEA-GAN	0.5901
gEA-GAN	0.5772
dEA-GAN	0.5721
TPN-gEA-GAN	0,5695
DM-gEA-GAN	n/a

6.2.3 Experimental Results Analysis

The DM-gEA-GAN results are not comparable to the others as the training was on only 500 epochs compared to 1000 for the others. The results that we obtain with DM-gEA-GAN are reasonably good though given that information. In my opinion, given the ability to predict t3 with only t2-t1 and the time difference ratio means, that this ”evolution” information is vital. All model should take as input at least two images at different time points.
The overall ranking of all the other models is similar to what we visually assessed in the last section, whith gEA-GAN and TPN-GAN being the best overall, and TPN-gea-GAN sacrificing visual quality and brain reconstruction to a better growth prediction. dEA-GAN is slightly below and simple pix2pix apparently the worst (which is normal as it was the simplest and quickest to train). In terms of score ratio/training time, gEA-GAN is the obvious winner as both dEA-GAN and TPN-gEA-GAN take at least 50% longer to train. 64-gEA-GAN has an easier task and achieves obviously better results for that reason.

Chapter 7

Discussion

7.1 On brain masking and coregistration

As the project went on I realized that Brain Masking and Coregistration were the two most vital steps of the project. Most obvious problems I encountered were due to extremely bright edges following poor brain masking, or brain at an ulterior date that looked completely different to brains at the first date. A dataset that is not properly preprocessed is completely useless, and as little as 10% images that are badly registered or masked can drastically reduce the quality of the test outputs.

7.2 On predicting 128x128x128 images ...

Predicting 3D 128x128x128 MRI actually means predicting more than two million pixels as opposed to around 60k for 256x256 slices. It means that training a gEA/dEA GAN model on 1000 epochs takes 24h and training the TPN-gEA takes twice as long, even with only 36 patients and parallelizing the task as much as possible given the GPUs of the remote machine. This is the reason why there are no hyper-parameters tests in this report. Obtaining comparable results when varying hyper-parameters would have taken forever. So I did informal comparisons in early epochs and more or less trusted the default values of the original code to choose my hyper-parameters, in order to focus on the models. For that reason I do not claim that the hyper parameters I chose are the very best for every model. But they are reasonably good and are the same across the tests with different models.

7.3 ...With a 36-patients dataset

Deep Learning is known to require vast amounts of data to be able to output high visual quality content without using overfitting to fake it, and conditional GAN are no different. I realized that over a thousand epochs the network became able to predict images that looked very similar to the target in the training set but when confronted to test patients never seen before, the predictions were mediocre.

Extracting a precise vision of what each different lobe should look like, the size of the ventricles, and how a tumour should grow, seems to be an extremely hard task in that regard. I do not think that changing the model (given this data and the way I preprocessed it) would be able to drastically improve the quality of the prediction.

Predicting longitudinal slices meant that for each scan in the initial dataset there were about fifteen slices that displayed a useful view of the brain and its structure (most scans are 256x256x25 initially, and of the 25 potential slices, around 15 display the brain clearly). That means in effect multiplying the dataset size by 15.
Moreover, if slices from all the patients were mixed together in the training and test set, some brain present in the test set had probably already been seen in the training set.

7.4 Edge Aware GAN for 2D slices

As I had the well processed slice dataset prepared by Andreas, and he provided a very clear README to train his models, I was at some point tempted to start from there and adapt the 3D edge aware GANs github implementation from "https://github.com/by-lab/Ea-GANs" to the slices dataset. But it revealed to be more work than what I was willing to invest in a 2D model. Indeed the Edge Aware GAN model has an architecture especially optimized for 128^3 brain images that I did not found very easy to efficiently carry to a 2D environment. I switched to preparing a 3D dataset.
If obtaining a bigger and better preprocessed dataset was not possible, this is the lead I would follow first if I was to continue the project

7.5 Ideas for new 3D models

7.5.1 Multimodal GAN

I looked into the GAN multimodal images concatenated to see if it was possible to extract information from T1, T2, and FLAIR and not only FLAIR. I was unable to solve several issues. For instance how to coregister all the different images together. Then how to combine them all efficiently in one image. As the tumour part is very visible in t2-flair and not so for other modalities I decided to give up on that lead. But I still think that with some work done on coregistering them together and maybe using an architecture that takes several images as inputs it would be very interesting to extract information from images of different modalities. Of course it would require high computational power to train such a big network in reasonable time.

7.5.2 Multi Scale Model

Esla gave this idea which is the reason why I trained the gEA-GAN in dimension 64x64x64, but I did not implement the final complete model. The idea is that, realizing how long it takes to train 1000 epochs for a 128x128x128 model, and that 128 is not even enough to match the resolution of the real MRI, it could be interesting to first train a model to predict a 64x64x64 image, that would be taken as input along the standard inputs of the 128x128x128 gEA-GAN. And if it works well, even feeding the 128x128x128 output to a 256x256x256 gEA-GAN. As the genereator in gEA-GAN is already a U-net that does that in a way, I am not sure it would be efficient, nonetheless it is a interesting lead.

Chapter 8

Conclusion

The task of predicting tumoral growth is a very hard one: an MRI features limited information, and the causes that determine tumour growth rate and the directions in which the tumour can grow may be hidden. It is already hard for a doctor with good knowledge of a patient, abstraction and analysis skills.

In terms of Artificial Intelligence and Computer Science it is yet another problem as it is the aggregate of many complex subtasks, the main goal being to generalize properly in high definition how a tumour grows from one or two images.

First, we need to analyse all the information that a AI model should get in order to perform the prediction. One image is probably not enough to predict a future image taken at an ulterior unknown date. So we added a time predictor to our model. But we cannot ignore that there can be different speeds of growth. So we added difference maps coupled with time difference as input for the model to learn the evolution. There is a trade-off between the complexity of the model (along with the time required to train it) and the information we want it to have.

Then, when we have a model that works properly it needs to be fed with a good amount of data, of good quality. Depending of how the data was collected initially and with which equipment, the masking and registration script need to be good enough and adapted to it.

And finally a machine must run the experiments. If the code is not optimized, training can easily take ten times as long as it should.

I have been confronted with all these issues. And I do believe that this project is a step in the right direction when it comes to automation of 3D tumour growth prediction. Especially DM-gea-GAN produced encouraging results and gEA-GAN as well as TPN-gEA-GAN were able to predict reasonable evolution for the tumours (with good overall image quality).

I also believe that the metric that we designed for this task could be used in other similar contexts. Being able to compare this score when using the different models showed something specific about each model and were coherent with our visual evaluation.

I am very happy and grateful to have been able to research in this area in which I sincerely believe.

I found the coding part very challenging, especially at the beginning due to the fact that I had to manage big pieces of codes written by other people (my experience in coding is 2-3 years of study). It helped me develop a lot of technical skills in that domain.

The research aspect is something I was used to as I had done a research project on Imputation (Statistical Analysis with missing data) last year. I enjoy it a lot. I find discovery tremendously exciting, even more when there is a whole community of researchers around who work on similar subjects. Reading papers helped me have ideas and motivation during the project.

Chapter 9

Appendix

9.1 Ressources

9.1.1 Machine

The machine that was used to conduct this project was a GPU machine located in the ITMAT Data Science group. It was accessible in ssh via the Imperial College VPN. So the models were trained on an Nvidia GeForce RTX 2080 Ti.

9.1.2 Tools and Frameworks

Python 3.7.8

Python is an interpreted, high-level, general-purpose programming language. Many tools and frameworks have been developed to support Machine Learning and Deep learning, which makes Python one of the most practical languages to work with as an AI Researcher. The implementation of Edge Aware GANs I was interested in was coded in Python. What's more, it the language I am the most proficient at.

Learn more at https://www.python.org

CUDA

CUDA is a parallel computing platform and API model created by Nvidia. The final code relies heavily on CUDA to make use of the full potential of the GPU machine.

Learn more at https://developer.nvidia.com/cuda-zone

Pytorch

PyTorch is an open source machine learning library for Python based on the Torch library. It provides tensor computing with strong acceleration via graphics processing units and a tape-based automatic differentiation system for Deep Neural Networks. The implementation of EA-GAN on Github uses Pytorch.

Learn more at https://pytorch.org/

SimpleITK

SimpleITK is a simplified, open-source interface to the Insight Segmentation and Registration Toolkit. The SimpleITK image analysis library is available in Python 3. I called on Simple ITK for the coregistration script as part of data preprocessing, and also to visualize 3D MRI on the remote Jupyter Notebook.

Learn more at https://simpleitk.org/

Numpy

NumPy is an essential library for Python. It supports large, multi-dimensional arrays and matrices, with a collection of mathematical functions to operate on these.

Learn more at https://numpy.org/

Matplotlib

Matplotlib is the most used plotting library for Python and NumPy.

Learn more at https://matplotlib.org/

Scikit Learn

Scikit Learn is machine learning library for Python. It offers various classification, regression and clustering algorithms that were used for computing some of the evaluation scores.

Learn more at https://scikit-learn.org/

Scikit Image

Scikit-image is an image processing library for Python. I used it to compute structural similarity and get the SSIM score.

Learn more at https://scikit-image.org/

Flirt

Flirt is an automated tool for linear brain image registration, both across the same and different modalities. It was used in the brain masking shell script.

Learn more at https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FLIRT

Visdom

Visdom is a tool to facilitate visualization of remote data. It focuses on supporting scientific experimentation. It is used in the original pix2pix implementation, and as a result helped me visualizing my results as well.

Learn more at https://github.com/facebookresearch/visdom.

9.2 Index of work credit

Credit for the work I used in the context of my project.

9.2.1 Performed by Kammy Pike

- Collection of the dataset we have been using and manual segmentation on a number of t1 images.

9.2.2 Performed by Elsa Angelini

- The shell script to perform brain masking **pipeline_mask_FLAIR_patients_v2.sh**

9.2.3 Performed by Andreas Zinonos

- The coregistration script **coregister.py**
- Some functions used in the **preprocessing.py** file designed to create pairs of 3D images. For instance the function that names each pair with the number of weeks that separates them.
- Time predictor model for slices (it helped me code my 3D time predictor)
- Pix2Pix Brain model which is basically Pix2Pix using Time predictor (it helped me code my TPN-gEA-GAN)

9.2.4 Performed by Biting Yu

Biting Yu implemented the gEA-GAN and dEA-GAN models, primarily to predict 128x128x128 images. This implementation relies heavily on Jun-Yan Zhu Pix2Pix implementation that Andreas used in his project.

- Models for gEA-GAN, and dEA-GAN
- DataLoader for datasets of paired images
- train and test scripts for them

9.2.5 Performed by myself

- **upsampling.py**, **rescaling.py** to apply these changes to all the eligible images of my dataset
- **mask_coregistering.py** to coregister the mask at the same time as the corresponding image
- 3D images pairing in **preprocessing.py**
- Slight changes to the data loader to take the time into account.
- Code for Time predictor model for 3D images
- Test and train scripts designed for Time Predictor
- Code for TPN-gEA-GAN model
- Code for DM-gEA-GAN model
- **eval.py** file to compute my new specific evaluation metrics
- Modification to **vizualizer.py** and **util.py** to add new ways of displaying the data (in particular to store and display sagittal and coronal views)
- Run of all the tests presented in this report

9.3 Legal and Ethics

Table 9.1: Ethics Checklist

	YES	NO
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		X
Does your project involve the use of human embryos?		X
Does your project involve the use of human foetal tissues / cells?		X
Section 2: HUMANS		
Does your project involve human participants?	X	
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues?		X
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?	X	
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?	X	
Does it involve processing of genetic information?		X
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		X
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?	X	
Section 5: ANIMALS		
Does your project involve animals?		X
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		X
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		X
Could the situation in the country put the individuals taking part in the project at risk?		X
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		X
Does your project deal with endangered fauna and/or flora/protected areas?		X
Does your project involve the use of elements that may cause harm to humans, including project staff?		X
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		X

	YES	NO
Section 8: DUAL USE		
Does your project have the potential for military applications?		X
Does your project have an exclusive civilian application focus?		X
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		X
Does your project affect current standards in military ethics e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		X
Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/ terrorist abuse?		X
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		X
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?		X
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		X
Section 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?	X	
Will your project use or produce goods or information for which there are data protection, or other legal implications?	X	
Section 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?		X

SECTION 2: HUMANS

Elsa Angelini, Matthiew Williams and I were the only Humans to participate to the project, but the private dataset itself consisted of MRI scans of patients diagnosed with brain tumours from the Charring Cross Hospital in London.

SECTION 4: PROTECTION OF PERSONAL DATA

Kammy Pike and Dr. Matthew Grech-Sollars at Charring Cross Hospital collected the dataset. The images only wear a number that relates them to a given patient, as well as sequence type, scan information (resolution and spacing), the scan dates, disease progression and procedure that took place before the scan (diagnosis, operation or chemo-radiotherapy). The data being private, I will not keep it on any personal device or release it publicly.

SECTION 10: LEGAL ISSUES WITH CODE

The code I have used along the project falls under the following categories and can be combination of:

- Public Python modules
- Public repositories from Github, namely "https://github.com/by-lab/Ea-GANs/" and "https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix"
- Scripts from Elsa Angelini and Andreas Zinonos
- Scripts from myself

This combined code is private on my Github, and will not be released publicly.

9.4 User Guide-Typical Session

Once on the remote machine:
Access my project: cd /data/MSc_students_accounts/mathieu/MscProject
Activate conda environment: conda activate brain_env
Launch visdom session: python -m visdom.server
Visualize the web image pages: python3 -m http.server 9999; index.html;

9.4.1 Preprocessing

Perform brain masking:
./pipeline_mask_FLAIR_patients_v2.sh /data/MSc_students_accounts/mathieu/MscProject/ brain_data/t2_masked_data /data/MSc_students_accounts/mathieu/MscProject/ brain_data/ t2_masked_data/BATCH3_20.02.19_nifti/DIGr_P044/04.10.17 20171004143249_t2_tirm_tra_dark-fluid_fs_3 nii
Coregistration: python3 coregister.py yourFolder
Pairing: python3 preprocessing.py yourFolder
Upsampling: python3 upsampling.py (change the data path inside the script)
Rescaling: python3 rescaling.py yourFolder
Applying a mask: fslmaths 'image_1.nii.gz' -mas 'mask.nii.gz' 'saving_location.nii.gz'

9.4.2 Training and Testing

cd Ea-GANs or cd Ea-GANs-time depending on the model you want to train.
Training:
python train.py –dataroot /data/MSc_students_accounts/mathieu/MscProject/brain_data /full_preprocessed/pairing –name example_gEaGAN21 –model gea_gan –which_model_netG unet_128 –which_direction AtoB –lambda_A 1.0 –dataset_mode aligned –use_dropout –batchSize 4 –niter 100 –niter_decay 100 –lambda_sobel 1.0 –fineSize 128 –lr 0.00002 –beta1 0.6

Testing:
python test.py –dataroot /data/MSc_students_accounts/mathieu/MscProject/brain_data/full_preprocessed/pairing –name example_gEaGAN20 –model gea_gan –which_direction AtoB –dataset_mode aligned –use_dropout

Training Time Predictor:
python train_time.py –dataroot /data/MSc_students_accounts/mathieu/MscProject/brain_data/full_preprocessed/pairing –name time_pred600 –model time_predictor –which_direction AtoB –lambda_A 1.0 –dataset_mode aligned_time –use_dropout –batchSize 1 –niter 100 –niter_decay 100 –lambda_sobel 1.0 –fineSize 128 –lr 0.00002 –beta1 0.65 (–gpu_ids -1)

Training gEA-TPN python train.py –dataroot /data/MSc_students_accounts/mathieu/MscProject/brain_data/full_preprocessed/pairing –name gea_TPN_test –TPN time_pred –model gea_TPN –which_model_netG unet_128 –which_direction AtoB –lambda_A 1.0 –dataset_mode aligned_TPN –use_dropout –batchSize 1 –lambda_sobel 1.0 –fineSize 128 –lr 0.00002 –beta1 0.65 –save_epoch_freq 10 –gamma 0.1 –niter 600 –niter_decay 200

9.4.3 Evaluation

python eval.py '/data/MSc_students_accounts/mathieu/MscProject/Ea-GANs-time/results/gea_TPN2/test_latest/images'

Bibliography

Agravat, R. R. and Raval, M. S. (2020). Brain tumor segmentation and survival prediction. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11992 LNCS:338–348. pages 4

Angelini, E., Clatz, O., Mandonnet, E., Konukoglu, E., Capelle, L., and Duffau, H. (2007). Glioma Dynamics and Computational Models: A Review of Segmentation, Registration, and In Silico Growth Algorithms and their Clinical Applications. *Current Medical Imaging Reviews*, 3(4):262–276. pages 4

Du, S., Guo, H., and Simpson, A. (2019). Self-Driving Car Steering Angle Prediction Based on Image Recognition. *arXiv:1912.05440*. pages 6, 14

Ezhov, I., Lipkova, J., Shit, S., Kofler, F., Collomb, N., Lemasson, B., Barbier, E., and Menze, B. (2019). Neural Parameters Estimation for Brain Tumor Growth Modeling. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11765 LNCS:787–795. pages 6

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, 3(January):2672–2680. pages 7, 8, 13

Ius, T., Angelini, E., Thiebaut de Schotten, M., Mandonnet, E., and Duffau, H. (2011). Evidence for potentials and limitations of brain plasticity using an atlas of functional resectability of WHO grade II gliomas: Towards a "minimal common brain". *NeuroImage*, 56(3):992–1000. pages 5

Jionghao, W. (2019). UU-Nets Connecting Discriminator and Generator for Image to Image Translation. *arXiv:1904.02675*. pages 9, 10, 11

Kaiser, B. and Albarqouni, S. (2019). MRI to CT Translation with GANs. *arXiv:1901.05259*. pages 1, 6

Kamnitsas, K., Ferrante, E., Parisot, S., Ledig, C., Nori, A. V., Criminisi, A., Rueckert, D., and Glocker, B. (2016). DeepMedic for brain tumor segmentation. In *MICCAI Brain Lesion Workshop*. pages 4, 53

Konukoglu, E., Clatz, O., Menze, B. H., Stieltjes, B., Weber, M. A., Mandonnet, E., Delingette, H., and Ayache, N. (2010). Image guided personalization of reaction-diffusion type tumor growth models using modified anisotropic eikonal equations. *IEEE Transactions on Medical Imaging*, 29(1):77–95. pages 5

Li, Y., Jia, F., and Qin, J. (2016). Brain tumor segmentation from multimodal magnetic resonance images via sparse representation. *Artificial Intelligence in Medicine*, 73:1–13. pages 3

Lucic, M., Kurach, K., Michalski, M., Bousquet, O., and Gelly, S. (2018). Are Gans created equal? A large-scale study. *Advances in Neural Information Processing Systems*, 2018-Decem(Nips):700–709. pages 15, 16

Pope, W. B. and Brandal, G. (2018). Conventional and advanced magnetic resonance imaging in patients with high-grade glioma. pages 2

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9351:234–241. pages 9

Shin, H. C., Tenenholtz, N. A., Rogers, J. K., Schwarz, C. G., Senjem, M. L., Gunter, J. L., Andriole, K. P., and Michalski, M. (2018). Medical image synthesis for data augmentation and anonymization using generative adversarial networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11037 LNCS:1–11. pages 6

Shmelkov, K., Schmid, C., Alahari, K., Shmelkov, K., Schmid, C., Alahari, K., and Eccv, G. A. N. (2018). How good is my GAN ? To cite this version : HAL Id : hal-01850447. pages 218–234. pages 16, 17

Taylor, G. W., Romero, A., Pineda, L., and Drozdzal, M. (2019). On the Evaluation of Conditional Gans. pages 53

Theis, L., Van Den Oord, A., and Bethge, M. (2016). A note on the evaluation of generative models. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pages 1–10. pages 15

Yi, X., Walia, E., and Babyn, P. (2019). Generative adversarial network in medical imaging: A review. *Medical Image Analysis*, 58. pages 8, 9

Yu, B., Zhou, L., Wang, L., Shi, Y., Fripp, J., and Bourgeat, P. (2019). Ea-GANs: Edge-Aware Generative Adversarial Networks for Cross-Modality MR Image Synthesis. *IEEE Transactions on Medical Imaging*, 38(7):1750–1762. pages 11, 12, 14, 20