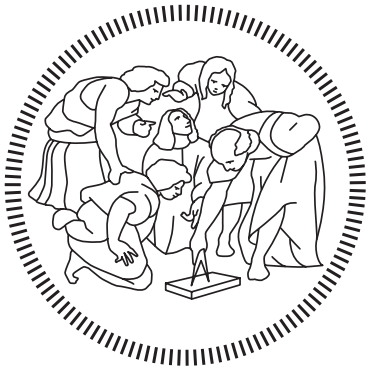
Politecnico di Milano

SCHOOL OF INDUSTRIAL AND INFORMATION ENGINEERING
Master of Science in Computer Science and Engineering



Brain Magnetic Resonance Imaging Generation using Generative Adversarial Networks

Supervisor
Daniele LOIACONO

Co-Supervisor
Edoardo GIACOMELLO

Candidate
Emanuel ALOGNA – 893592

Academic Year 2018 – 2019

To the ones that never stop believing in themselves.

To the fools who always dream.

Ringraziamenti

Prima di tutto vorrei ringraziare il Prof. Daniele Loiacono, mio relatore e guida in questo percorso non privo di difficoltà, e il Dott. Edoardo Giacomello per la loro grande disponibilità durante questi mesi di lavoro ma anche per la pazienza mostrata, il continuo supporto, gli spunti fondamentali e i confronti costruttivi che mi hanno permesso di realizzare questa tesi. Grazie per avermi stimolato e permesso di appassionarmi ancora di più a questa materia.

Grazie a mia mamma e mio papà, che non hanno mai smesso di incoraggiarmi e mi hanno supportato (e sopportato) in ogni modo possibile durante questi anni, permettendomi di concentrarmi sullo studio a tempo pieno e dandomi la possibilità di vivere una bellissima esperienza all'estero. Tutto questo, senza di loro, non sarebbe stato possibile.

Un ringraziamento speciale ai miei fratelli che mi spingono a mettermi sempre in gioco: a Dani, la persona con la quale ho sempre potuto parlare di qualsiasi cosa, e a Mirko che, nonostante la distanza, non ha mai fatto mancare i suoi preziosi consigli. Grazie a Simona, che ormai è come una sorella, e grazie a nonna Uccia, che aspettava più di chiunque altro questo momento.

Ringrazio Liuch, collega nonchè uno dei miei migliori amici, con cui ho avuto la fortuna di condividere questi cinque anni universitari, un progetto dopo l'altro, esame dopo esame. Finalmente siamo riusciti a tagliare questo traguardo.
Un grazie speciale anche a Ippo, Rich, Erik e Barte per esserci sempre stati, anche nei momenti difficili. Amici di una vita, con cui spero di condividere ancora tanti momenti come questo.

Desidero inoltre ringraziare Lollo e Lello, compagni di studio in biblioteca fino a tarda notte, e grazie a Gio, per le domeniche passate in aula studio sin dalla mattina e per le sciate in montagna in uno dei momenti di maggiore stress di questi mesi. Grazie infine a Salva, Dave, Mutto e Fabio che, tra calcetti e pause in biblioteca, hanno reso più leggeri gli ultimi anni di questo viaggio.

Contents

List of Figures

List of Tables

Abstract

Magnetic Resonance Imaging (MRI) is nowadays one of the most common medical imaging techniques, due to the non-invasive nature of this type of scan that can acquire many modalities (or sequences), each one with a different image appearance and unique insights about a particular disease. However, it is not always possible to obtain all the sequences required, due to many issues such as prohibitive scan times or allergies to contrast agents. To overcome this problem and thanks to the recent improvements in Deep Learning, in the last few years researchers have been studying the applicability of Generative Adversarial Network (GAN) to synthesize the missing modalities. Our work proposes a detailed study that aims to demonstrate the power of GANs in generating realistic MRI scans of brain tumors through the implementation of different models. We trained in particular two kind of networks which differ from the number of sequences received in input, using a dataset composed by 274 different volumes from subjects with brain tumor, and, among a set of different evaluation metrics implemented to test our results, we validated the quality of the predicted images using also a segmentation model. In addition, we analysed the GANs trained by performing some experiments to understand how the information passes through the generator network. Our results show that the synthesized sequences are highly accurate, realistic and in some cases indistinguishable from true brain slices of the dataset, highlighting the advantage of multi-modal models that, compared to the unimodal ones, can exploit the correlation between available sequences. Moreover, they demonstrate the effectiveness of skip connections and their crucial role in the generative process by showing the significant degradation in the performances, analysed in both a qualitative and quantitative way, when these channels are turned off or perturbed.

Estratto in lingua Italiana

L'imaging a risonanza magnetica (MRI) è oggi una delle più comuni tecniche di generazione di immagini mediche, per via della natura non invasiva di questo tipo di scansione che permette di acquisire varie modalità (o sequenze) della parte del corpo scansionata, ognuna diversa dall'altra per livello di risoluzione e contrasto utilizzato. Tuttavia, non è sempre possibile ottenere tutte le sequenze richieste, a causa di molteplici problemi, tra cui i tempi di scansione proibitivi o le allergie dei pazienti agli agenti di contrasto. Per ovviare a questo problema e grazie ai recenti miglioramenti nel campo del Deep Learning, negli ultimi anni i ricercatori hanno studiato l'applicabilità delle Reti Antagoniste Generative (GAN) alla generazione delle modalità mancanti. Il nostro lavoro propone uno studio dettagliato che mira a dimostrare l'abilità delle GAN nel generare scansioni MRI realistiche di tumori cerebrali attraverso l'implementazione di diversi modelli. Abbiamo allenato in particolare due tipi di reti che differiscono per il numero di sequenze ricevute in input, utilizzando un dataset composto da 274 diversi volumi appartenenti a pazienti con tumori cerebrali e, tra una serie di diverse metriche implementate per valutare i nostri risultati, abbiamo validato la qualità dell'immagine generata dalla rete usando anche un modello di segmentazione. Inoltre, abbiamo analizzato le GAN addestrate, eseguendo alcuni esperimenti per capire come il contenuto informativo ricevuto in input passi attraverso il generatore, ovvero una delle due reti neurali che compongono una GAN. I nostri risultati dimostrano che le sequenze sintetizzate sono altamente accurate, realistiche e in alcuni casi indistinguibili dalle immagini provenienti dal dataset, evidenziando il vantaggio dei modelli multi-input che, rispetto a quelli single-input, possono sfruttare la correlazione presente tra le sequenze che sono disponibili. Inoltre, dimostrano l'efficacia delle skip connections e il loro ruolo fondamentale nel processo generativo mostrando come, spegnendo o perturbando i canali, le prestazioni della rete subiscano un calo significativo.

Chapter 1

Introduction

Medical imaging is crucial for clinical analysis and medical intervention since it gives important insights about some diseases whose structures might be hidden by the skin or by the bones. One of the most common techniques used nowadays is the Magnetic Resonance Imaging (MRI), ubiquitous in hospitals and medical centers, first because of its non-invasive nature, since, differently from other imaging technologies, doesn't make use of X-ray radiography and secondly because of the recent improvements in the software and hardware instrumentation used. In this type of imaging, various sequences (or modalities) can be acquired and each sequence can give useful and different insights about a particular problem of the patient. For example the T_1-weighted sequence can distinguish between gray and white matter tissues while T_2-weighted is more indicated to highlight fluid from cortical issue.

The problem about MRI is that sometimes there isn't the possibility to acquire all the sequences that would be required in order to diagnose a disease and this is due to different reasons: sometimes there isn't enough time to collect all the needed sequences, sometimes it's too expensive to acquire all of them. Furthermore, missing sequences might occur because of allergies in the patient that don't allow to obtain modalities where an exogenous contrast agent, a substance useful to increase the contrast between structures or fluids within the target area, is used to make the image clearer. It can also happen that, for a given patient, some scans might be unusable due to the presence of errors, corruptions or machine settings not defined in the proper way [1].

It would be useful, then, to find a way to generate a prediction of the missing sequences by using the modalities already acquired for a given patient. These generated predictions then could be used either as direct aid to the doctor that has to make a diagnosis for a patient that maybe can't have injected into his body any kind of

contrast agent, or could be used as one of the input pieces of a bigger pipeline.

Thanks to the recent improvements in Machine Learning, but more in particular in the Deep Learning (DL) field, the generation of missing modalities has become an objective feasible to reach, both in terms of efficiency of the obtained result and in terms of time spent in order to reach something meaningful.

In the last few years interest toward Machine Learning and Deep Learning has grown exponentially: just to give an idea to the incredible spread that is having this Artificial Intelligence (AI) research area, in the 2016 and 2017 over 400 contributions related to Deep Learning applied to Medical Image Analysis were published [2]. It has been shown that DL has been able to reach very competitive results in many medical task: classification, detection, segmentation, registration of different areas and structures within the human body. Many progresses have been done and further are yet to come in order to reach results as accurate and precise as possible in crucial applications such as cancer cell classification, lesion detection, organ segmentation and image enhancement [3].

The most successful DL architecture in medical image analysis is the Convolutional Neural Network (CNN): a neural network with multiple hidden layers between the input and the output layers. The initial problem with this type of architecture was that researchers weren't able to train these deep neural networks in an efficient way. A breakthrough was the work described in [4] that represents a huge contribution to the field since the proposed CNN, called AlexNet, won the ImageNet competition in 2012 by a large margin. The peculiarity of AlexNet was that its competitive results and high performances were obtained in relative short time, due to the fact that the designer of this network made the training feasible by the utilization of a Graphics processing unit (GPU).

Another important breakthrough in the field was represented by the introduction of Generative Adversarial Networks, a class of machine learning systems invented by Ian J. GoodFellow in 2014 [5] that we used in this work to solve the problem of missing modalities.

1.1 Scope

In this work we study the applicability of Generative Adversarial Networks to the domain of medical imaging, by focusing on the generation of brain MRI sequences

in order to overcome the problem of missing or unusable modalities and so to avoid repeating the exams multiple times. In particular, we propose a detailed study on the predictive power of three different GAN implemented: the first one is a unimodal model proposed by Isola et al. and called pix2pix [6], while the other two networks are MI-pix2pix, a multi-input version of pix2pix, and MI-GAN, a modified version of the MM-GAN proposed by Sharma et al. in [1], adapted to the multi-input single-output scenario.

We evaluate, then, the results obtained from the trained networks by assessing the performances in both a qualitative way, so with human perception, and a quantitative way, through different metrics implemented to test the generated quality of the tumor area, that needs to be synthesized with high accuracy, as well as the overall quality of the whole image.

Moreover, for studying the capabilities of the network and how skip connections affect the generative process, we show how much the performances can deteriorate by switching off or perturbing the channels in the generator network.

The two main contributions of this work can be summarized as follows:

- We define and compare two multi-modal GANs, MI-pix2pix and MI-GAN, that receive the same number of inputs but have different architectures and use different loss functions.
- We present a systematic analysis of skip connections that extends the analysis of the generator architecture in [6].

1.2 Thesis Structure

In Chapter 2 we present the related work and the theoretical background needed to make more understandable the work discussed in the following chapters. Chapter 3 describes in detail the dataset used and how it has been preprocessed in order to make it work with our models. In Chapter 4 we present to the reader the neural network architecture implemented, along with the training algorithm and the metrics that we decided to use in order to evaluate, quantitatively, the results obtained, that are presented in Chapter 5. In Chapter 6 further experiments on our trained models are described. Finally, Chapter 7 contains our general considerations about this work, a description of the open problems related to the topic and a discussion on possible applications and future develops.

Chapter 2

State of the Art

In this chapter we first present the related work on the Generative Adversarial Networks in the literature (2.1) and then, in Section 2.2, we introduce the theoretical aspects our work is based upon, while giving the reader an overview of the tools and techniques we applied in our models, in order to better understand the setting in which our work takes place.

2.1 Related Work

GANs have been used in many data domains, but the images one is where this type of neural network obtained the most significant results so far. In [5] this architecture has been tested for the first time on a range of image datasets including MNIST [7], the Toronto Face Database [8] and CIFAR-10 [9]. Isola et al. applied GANs to the image-to-image translation task and, in order to explore the generality of their work [6], experimented with a variety of settings and datasets such as generating the night version of an image whose daily version was given as input to the network or synthesizing the picture of a building by only observing its relative architecture label as input after a training performed on CMP Facades (Figure 2.1).

2.1.1 GANs in Medical Imaging

Countless studies about medical image synthesis were approached using GANs. In particular, cross modality image synthesis (so the conversion of the input image of one modality to the output of another modality) is the most important application of this architecture: in 2018, was even published a review [2] about all the work and progresses that have been done in the field of medical imaging through the application of GANs. The authors described the magnetic resonance as the most common imaging modality explored in the literature related to this generative approach invented by Goodfellow

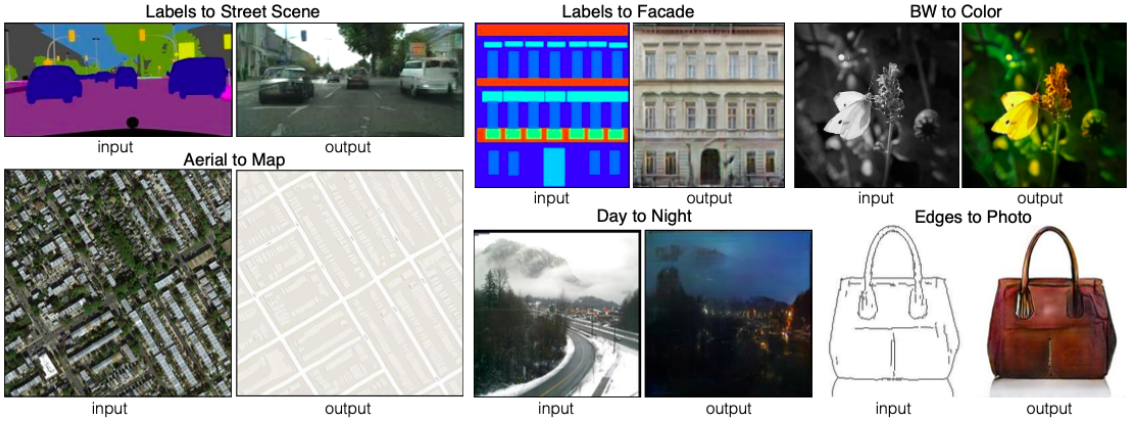


Figure 2.1. Examples of image-to-image translation from the work of Isola et al. [6]

and this is probably due to the fact that GANs, through the cross modality synthesis, can reduce the excessive amount of time requested from Magnetic Resonance (MR) acquisition.

Many different approaches and datasets have been used in the literature in order to overcome the problem of missing modalities. Orbes-Arteaga et al., in [10], since most of the datasets contain only T_1 or T_1/T_2/PD scans due to logistical reasons, implemented a GAN that generates T_{2flair} of the brain using T_1 modality. In [11], Camileri et al. developed a variant of the original GAN, called Laplacian pyramid framework (LAPGAN) that synthesizes images in a coarse-to-fine way. This method, as the name suggests, is based on Laplacian pyramid and allows to generate initially an image with low resolution and then, incrementally, add details to it. Another approach to generate missing modalities was proposed in [12] where they presented two possible scenarios, based on the given dataset: they use a model called pGAN (that incorporates a pixel-wise loss into the objective function) when the multi-contrast images are spatially registered while they adopt a cycleGAN [13] in the more realistic scenario in which pixels are not aligned between contrasts. A cycleGAN is a GAN variant characterized by the fact that has two generators, two discriminators and uses a cycle consistency loss with a collection of training images from the source and target domain that don't need to be related in any way.

Most of the papers about Generative Adversarial Networks, and also this work, though, are based on a training performed using paired examples, with input image and target image perfectly aligned.

It is worth to cite also the work done by Anmol Sharma and Ghassan Hamarneh that were, to the best of their knowledge [1], the first to propose a many to many generative model, capable of synthesize multiple missing sequences given a combination

of various input sequences. Furthermore, they were the first to apply the concept of curriculum learning, based on the variation of the difficulty of the examples that are shown during the network training.

MRI isn't the only imaging technique where GAN has been applied to: in literature is possible to find various publications of generative models that use Positron Emission Tomography (PET), Computerized Tomography (CT) or Magnetic Resonance Angiohraphy (MRA) images. Olut et al., in 2018, demonstrated that GAN works efficiently even when the source imaging technique is different from the target one: they were the first ones to synthesize MRA brain images from T_1 and T_2 MRI modalities [14]. Ben-Cohen et al. published in the same year an article [15] explaining how to generate PET images using CT scans through a fully connected neural network, whose output is improved and refined by a conditional GAN (cGAN) [16].

The work related to the generation of medical image is huge and new papers about the topic are being published every week. Many approaches have been already tested and many datasets have been used but there are still lots of challenges that need to be solved in order to be able to be employed in medical imaging [2] and since there is still room for improvements in the application of generative models in medical imaging, we present in this work a complete study on different architectures of GAN used to generate missing modalities of brain scans, followed by an analysis of the information that passes through the inner channels and through the skip connections of the generator networks.

2.2 Theoretical Background

The architecture of the main model used in our work, the Generative Adversarial Network, is based on the Convolutional Neural Network: a deep neural network with many hidden layers that nowadays is proved successful in many application going from Image Recognition to Video Analysis, passing through Natural Language Processing, Anomaly detection, Drug Discovery, etc.

2.2.1 Convolutional Neural Network

The 1980, period in which David H. Hubel and Torsten Wiesel gave crucial insights about the structure of the visual cortex (the researchers are winners of the 1981 Nobel Price in Physiology or Medicine for their work), is the starting point from which *convolutional neural networks* started to gradually evolve into what is today

the architecture most used in DL.

The milestone is represented by the work of Yann LeCun that in 1998 developed a model, LeNet-5, to identify handwritten digits for zip code recognition in the postal service [7]. This model was composed by traditional building blocks such as fully connected layers and sigmoid activation functions but also by convolutional layers and pooling layers, introduced for the first time.

The most important building block is the convolutional layer, based on the mathematical operation from which CNN take its name. This layer allows to extract features by convolving the input with a certain number of filters, producing a stack of feature maps, one per each filter that was convolved with the input image. Each filter basically tries to capture, in the first layers of the network, low-details of the image by producing this feature map that highlights which are the areas on the input that were activated by the filter the most. In the the next hidden layers this small low-level features are assembled into higher-level features and this hierarchical structure is one of the reason why CNNs work so well and are so widely used [7].

2.2.2 Generative Adversarial Network

Generative Adversarial Network was proposed in 2014 by Ian J. GoodFellow [5] and represents a new framework for estimating generative models in an adversarial setting. The system is composed by two neural networks: a discriminator D, typically a CNN, and a generator G that are trained simultaneously.

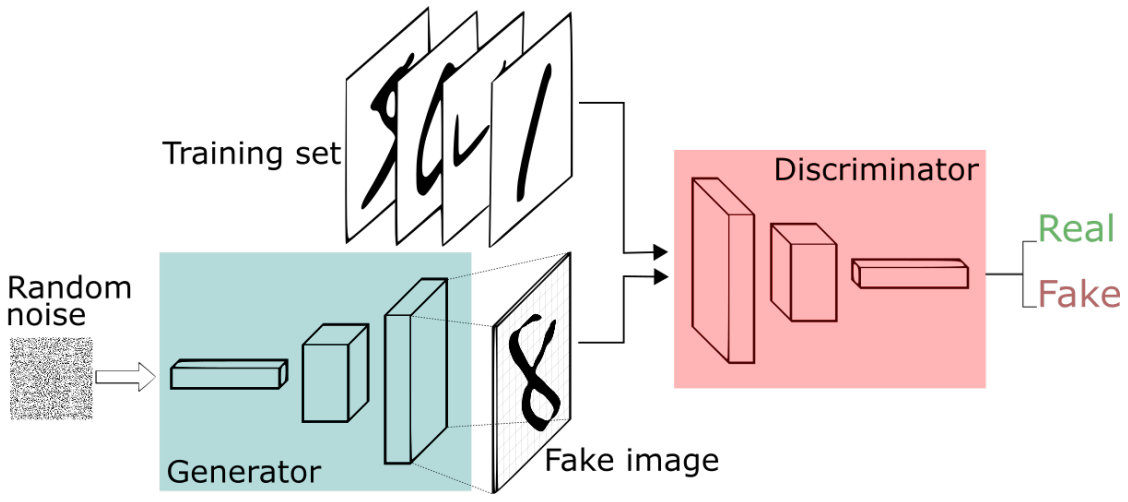


Figure 2.2. Generative Adversarial Network Framework [17].

In particular, G is trained to learn the probability distribution of the data given as input and to generate synthesized data that exhibits similar characteristics to the

authentic data while the discriminative model estimates the probability that a sample came from the training data rather than G.

This adversarial setting is usually explained through the example of the counterfeiter (generator) that tries to produce and use fake money without being caught from the police (discriminator) that, on the other hand, tries to detect the counterfeiter fake currency. The process continues until the counterfeiter becomes able to trick the police with success. GANs work in the same way, with the two networks that compete with each other in a game (corresponding to what is called minimax two-player game in Game Theory [18, p. 276]): the discriminator tries to distinguish true images, belonging to the input dataset, from fake images produced by the generator, whose objective is instead to learn to generate the most realistic data possible to be able to fool D.

Equation 2.1 shows the losses for D and G used in the original architecture [5].

$$\begin{aligned} L_D^i &\leftarrow \frac{1}{m}\sum_{i=1}^{m}[\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))] \\ L_G^i &\leftarrow \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(G(z^{(i)}))) \end{aligned} \tag{2.1}$$

where z^i is a batch of random noise and x^i is a batch of true m data samples. The two networks are trained alternately using *Backpropagation*, in such a way that the generator can learn to synthesize the images based on the discriminator feedback.

It's important to highlight that the G never sees real images: it just learns the gradients flowing back through the discriminator. The better the discriminator gets, the more information about real images is contained in these secondhand gradients, so the generator can make significant progress [19, p. 1301]. As the training advances, the GAN may end up in a situation that is called *Nash equilibrium*, so a situation in which the generator produces perfectly realistic images while the discriminator is forced to guess (50% real and 50% fake). Unfortunately, even training a GAN for a long time, there is no guarantee to reach this equilibrium [19, p. 1307].

2.2.3 Deep Convolutional GAN (DCGAN)

The problem with the original GAN paper [5] is that GoodFellow et al. experimented with convolutional layers but only with small images. In the following months researchers tried to work with deeper neural networks for larger images and, in 2015, Radfrod, Metz and Chintala presented an improved version of the original architecture with some tricks and variations [20].

The guidelines proposed, that we followed in our work, in order to obtain a stable training even with large images, are:

- Replacement of any pooling layers (downsampling layers that allow to reduce the spatial information of the image) with strided convolutions in the discriminator and with transposed convolutions (similar to the convolution layer but goes in the opposite direction by enlarging the spatial dimensions) in the generator.
- Batch normalization [21] - a normalization of the input computed across the whole batch - in every layer (with the exception of the generator output layer and the discriminator input layer).
- Remove fully connected hidden layers (whose neurons are connected to every node of the previous layer), both in G and D.
- ReLu as activation functions [22] in G for all layers except for the output, which uses tanh (hyperbolic tangent activation function with an output range from -1 to +1). ReLu is a non linear function that maps its input between 0 to infinity.
- use of Leaky ReLU: activation function similar to ReLU but with a small slope for negative values that avoid neurons to stop outputting anything other than a 0 value. It is used in D for all layers.

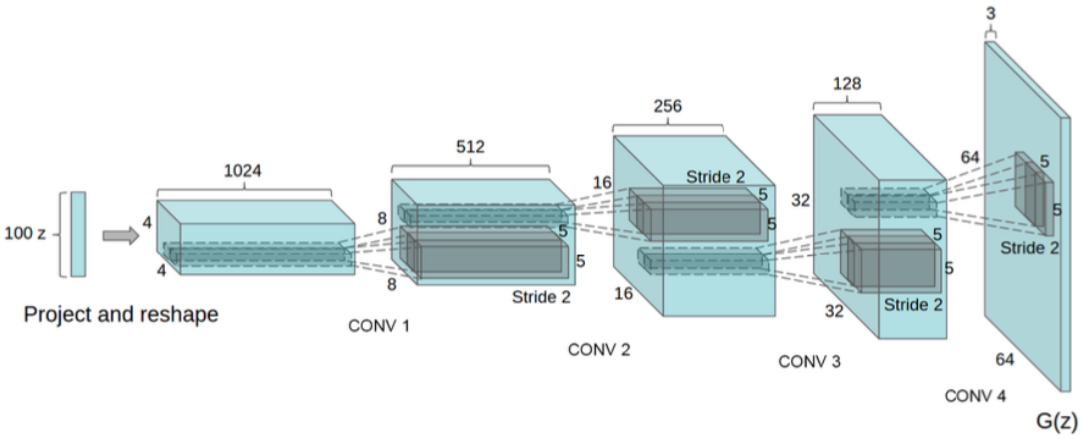


Figure 2.3. DCGAN generator used in the work of Radford, Metz and Chintala.

2.2.4 Conditional GAN

In the *Conclusions and future work* of the original GAN paper [16], GoodFellow et al. suggested that the proposed framework could have been extended by conditioning the input of both G and D. This extension was realized by Mehdi Mirza and Simon

Osindero that developed a Conditional Generative Adversarial Net. The authors explain that in a unconditioned generative setting there is no possibility to control the modes of the data that is generated, while, by conditioning the input with some extra information y, it's possible to direct the data generation process [16].

Such extra input can be any kind of auxiliary information, such as class labels or data coming from other modalities (in our work we conditioned the input with images as extra information), so the prior noise z is combined to y as input to the generator while the discriminator takes as input data x and y.

2.2.5 Image-to-Image translation with Conditional GAN

After the publication, in 2015, of *Conditional Adversarial Nets* [16], many researchers started to apply this kind of architecture to the *Image-to-Image translation*, the task of translating one possible representation of a scene into another one, using images as auxiliary information to control the data generation. An important contribute was given by Phillip Isola et al. that developed a general framework that is not application-specific and is generally known as Pix2Pix [6].

Pix2Pix works so well in many domains because of several architectural choices that were adopted both in the generator and in the discriminator.

Figure 2.4. Thermal images translated to RGB photos: an example of Image-to-Image translation using Pix2Pix [6].

None of these architectural choices were introduced for the first time by the authors of Pix2Pix: they have been already explored in many papers related to GAN and focused on specific applications, but for the first time they have been used for a general-purpose solution to image-to-image translation. The neural networks chosen by Isola et al. are the same ones that have been used in our work: a "U-Net"-based architecture [23] as generator and a "PatchGAN" classifier, similar to the one proposed in [24], as discriminator.

Furthermore, the authors found beneficial to mix the GAN objective to a more traditional loss such as the L1 loss. So, in addition to the objective of a conditional

GAN, that can be expressed as

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))], \tag{2.2}$$

where G tries to maximize the objective against the adversarial D that tries to minimize it, a L1 loss was used in order to measure the distance between the ground truth image and the generated image:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]. \tag{2.3}$$

L1 loss is preferred to the L2 one, since it produces results less blurred. The final objective, then, is:

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G). \tag{2.4}$$

Pix2pix is the model whose performances are used as baseline in our work and, because of this, we present below further details on the U-Net architecture and on the PatchGAN architecture, in order to make more understable the next chapters.

U-Net

Pix2pix adopts as generator a U-Net architecture [23], that allows to obtain better results with respect to the ones reached by generators composed by an encoder-decoder network (Figure 2.5).

The important problem in image-to-image translation, so in a setting where we need to map a high resolution input grid to a high resolution output grid, by maintaining the same underlying structure between an input and output image that differ in surface appearance [6], is that, using an encoder-decoder network, the input image is simply downsampled progressively until a bottleneck layer, after which the process is reversed and the information passes through many upsample layers.

Because of this bottleneck layer, it would be preferable to have to shuttle all this information directly across the net: this is obtained adding to the network some links, called **skip connections** between, for example, the i layer and the n-i layer, where n is the total number of layers.

Skip connections result very useful in tasks such as image colorization, where input and output share the location of the prominent pixels, and are able to prevent the problem of losing information through the bottleneck.

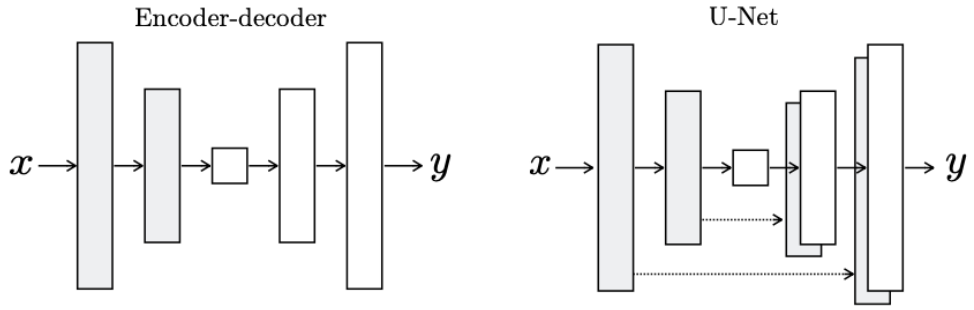


Figure 2.5. Two choices for the generator architecture. U-Net has skip connections between mirrored layers [6].

PatchGAN

Isola et al. explain that, since L1 loss (Eqn. 2.3) is able to capture accurately low-level frequencies, it's sufficient to restrict the GAN discriminator focus only on the high frequencies. High frequencies can be captured by using a CNN, called *Patch*GAN [24], that is able to discriminate small local *NxN* patches of the images and assign to these a fake or true label. Averaging all the responses across the image provides then the ultimate output of D [6].

The authors demonstrate, after testing various kinds of patches, such as a 1x1 "PixelGAN" and a 286x286 "ImageGAN", that a 70x70 PatchGAN gives the best results in terms of blurriness, sharpness, general quality of the image and presence of any artifacts (Fig.2.6)

Figure 2.6. Path size variations going from 1x1 "PixelGAN" to "286x286" "ImageGAN" passing through the 70x70 PatchGAN that produces sharp results in both the spatial and spectral (colorfulness) dimensions [6].

Furthermore, scaling the patch beyond 70x70 doesn't improve the quality of the output and has the downside of a longer training needed, because of the larger number of parameters.

2.3 Summary

In this chapter we presented the solution proposed in the last years, thanks to recent improvements in DL, to the problem of missing modalities.

After this, we described the main architectures upon which is based our work, starting from the original GAN and going through all the theory behind generative adversarial networks, giving to the reader a solid background about Deep Convolutional GAN, Conditional GAN and the two networks that compose Pix2Pix: U-Net as generator and PatchGAN as discriminator. Theory that is needed in order to understand the next chapters and in particular chapter 4 that shows how we implemented the network architecture in our system.

Chapter 3

Brain Tumor MRI

In this chapter we first give some details about Magnetic Resonance Imaging (Section 3.1) and then we discuss about the problem of missing modalities, with a particular focus on brain tumor imaging (Section 3.2). We also describe the data used in our experiments, by presenting details of the dataset: in Section 3.3 we illustrate how it is organized and which modalities contains while Section 3.4 gives a detailed description about the preprocessing steps applied in order to prepare the images before being used in our models.

3.1 Magnetic Resonance Imaging

We already discussed, in Chapter 1, why Magnetic Resonance Imaging (MRI) nowadays is so important and so widely used in every hospital and medical center: first of all it's a non-invasive procedure, since it doesn't make use neither of X-ray nor of ionizing radiation.

In order to scan a portion of the body radio waves are used: they interact with specific molecules in the body (protons, the nuclei of hydrogen atoms) and these radio signals are turned on and off. The energy in the waves is absorbed by the atoms in the target area and then reflected back, through the tissues, out of the body: when this happens, the signals coming out are captured by the MRI machine. Finally these captured signals are sent to the MRI computer and combined together in a 3D image.

This medical imaging technique has also the benefit of being extremely clear in details belonging to soft-tissues. Furthermore, it can scan larger parts of the body with respect to other imaging techniques, can produce hundreds of scans from almost any direction and in any orientation and the contrast agents used to obtain some particular sequences are less likely to cause some allergic reactions that may occur when iodine-based substances are used, for example for X-rays and CT [25].

What makes very interesting this imaging technique is that a single MRI scan is a grouping of multiple pulse sequences (modalities), each one highlighting different tissue contrast views and spatial resolution, allowing to give to the doctor various insights about a possible disease, since each modality presents a particular, and sometimes unique, image appearance, that is not possible to observe in other modalities: a clear example of it is the T_2-fluid-attenuant inversion recovery (T_{2flair}), used in brain imaging. This modality is very similar to T_2-weighted (T_2) beside the fact that, as the name suggests, the Cerebrospinal fluid (CSF) effect on the image is suppressed and allows to see clearly the tumor area. Figure 3.1 shows two brain MRI sequences belonging to the same patient, from the dataset BRATS2015, where is possible to see how the sequence on the right, T_{2flair}, is able to highlight the tumor (white color) in a better way than the sequence on the left, T_2, by suppressing the CSF: in order to obtain the scan on the right, the ventricular CSF signal is dampened and this causes the highest signals, belonging to tumors or other brain abnormalities, to have a light color, while the CSF appears black.

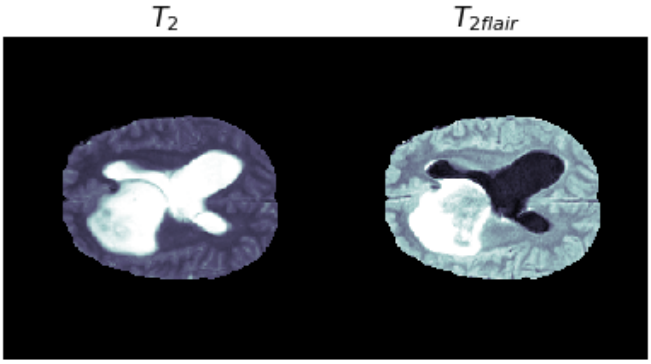


Figure 3.1. Brain MRI sequences - T_2 and T_{2flair} - of the same patient, from BRATS2015.

3.2 Brain MRI Generation

The problem of MRI is that often not all the sequences are available. Sometimes there isn't the possibility to acquire all the scans required for a diagnosis: prohibitive scan times, for instance, is one of the main issues in the magnetic resonance imaging acquisition. Moreover, modalities might be missing or unusable because of scan corruption, artifacts, incorrect machine settings but also due to high costs in terms of money to perform the screening. Another important issue is represented by contrast allergies: certain modalities can't be obtained from patients that can't have injected into their body a contrast agent.

Being able to generate realistic MRI scans would allow, first, to use these sequences

as a direct aid to the doctor that might need more information to diagnose a specific disease and, secondly, to use the new images as input of a downstream analysis tool such as a segmentation model that might require to receive all the sequences in order to segment, for example, a tumor and to distinguish between unhealthy cells from healthy ones.

A possible solution comes from the field of Deep Learning where a recent breakthrough opened the possibility of generating missing modalities: Generative Adversarial Networks, whose theory behind has been extensively described in Subsection 2.2.2. In this work we study the generative power of GANs applied to the MRI domain. In particular our focus is on brain tumor imaging (also known as neuroimaging): the incidence of brain tumors has been increasing in the last 20 years and it is one of the most common type of tumor in the world, according to an epidemiological review from the World Cancer Research Journal [26]. Because of this an early detection of the tumor, after generating all the sequences required for a diagnosis, as well as preventive measures to reduce the risk factors of this disease, are crucial.

3.3 Dataset

The dataset used in this work comes from the Multimodal Brain Tumor Segmentation Challenge 2015 (known as BRATS2015) [27,28]. Challenge with the purpose of focusing on the evaluation of state-of-the-art methods for the segmentation of multimodal MRI scans of the brain.

Even though BRATS2015 comes with a training and a test set, for our work we used only the information contained in the first set, since the training includes, per each scan, also the ground truth, so a segmented image of the tumor, that we used in order to measure, during the training but also during the evaluation phase, the quality of the synthesized brain image in the tumor area.

3.3.1 Dataset Organization

BRATS 2015 is composed by fully anonymized images of the Cancer Imaging Archive from 274 patients. In particular, there are 220 high grade subjects (HG) and 54 low grade subjects (LG): the purpose of a grading system for the tumors is to indicate the growth rate and how much is likely the spread into the whole brain.

Brain tumors are classified on a scale from 1 to 4: the ones corresponding to a spread level of 1 or 2 are considered LG, while the ones classified as level 3 or 4 are indicated as HG and are the most aggressive and malignant brain tumors [29].

Figure 3.2 shows a T_{1c} slice and the relative tumor area from two different patients:

the first one (above) is a high grade subject while the second one (below) belongs to a low grade subject. Looking at this example it is easy to understand that the tumor grade isn't just a measure of the overall dimensions: it's a measure of how aggressive is this disease based on the appearance of the cells under a microscope and how malignant they look (cells and the organization of the tissue of an high grade do not look like normal cells). [30].

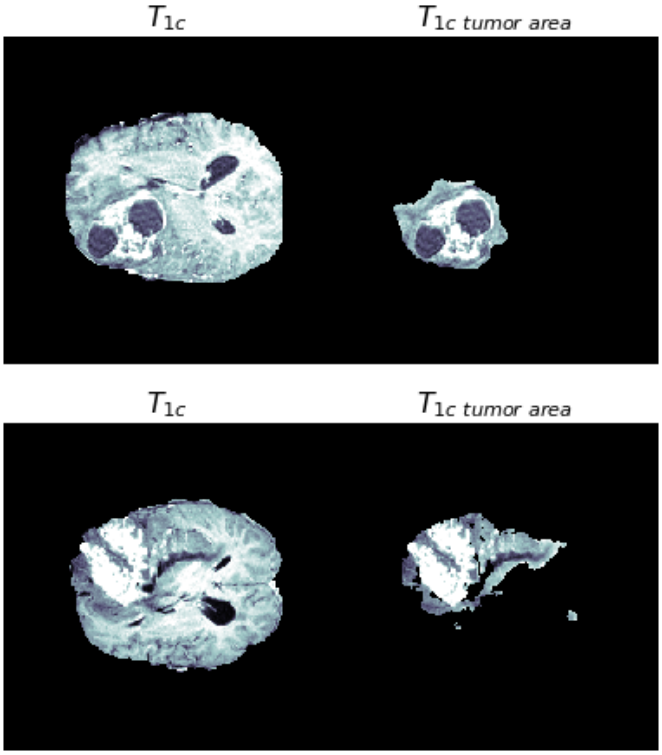


Figure 3.2. T_{1c} tumor slice in HG (above) and LG (below) subjects.

Since we believe it is important to maintain the balance between HG and LG subjects during training and evaluation phases, we applied stratified sampling, following the approach of [31], in the splitting of the dataset in three different sets: training (80% of the original dataset was assigned to this set), validation (10%) and test (10%) with 219 patients assigned to the first set, 27 to the validation one and 28 to the test.

Stratified sampling was used in order to show to the model, during training, the right amount of HG and LG without the risk of wrong generations once the model would have been used with the test set.

For example, let's suppose that a generator of MRI scans is trained with a set containing only LG tumors. Then, when it comes the time to test this model with a new set, with completely unseen images of HG and LG tumors, our trained model would probably be able to generate good results when it receives a LG patient as input, while it would produce wrong synthesized images, especially in the tumor area,

when it comes to generate the missing modality for a HG patient.

This explains why, in the splitting of the dataset, we chose to maintain a balance in the number of instances belonging to the two types of patient: stratified sampling allowed us to have HG and LG subjects represented with approximately equal proportions in all the three subsets, as it is shown in Figure 3.3.

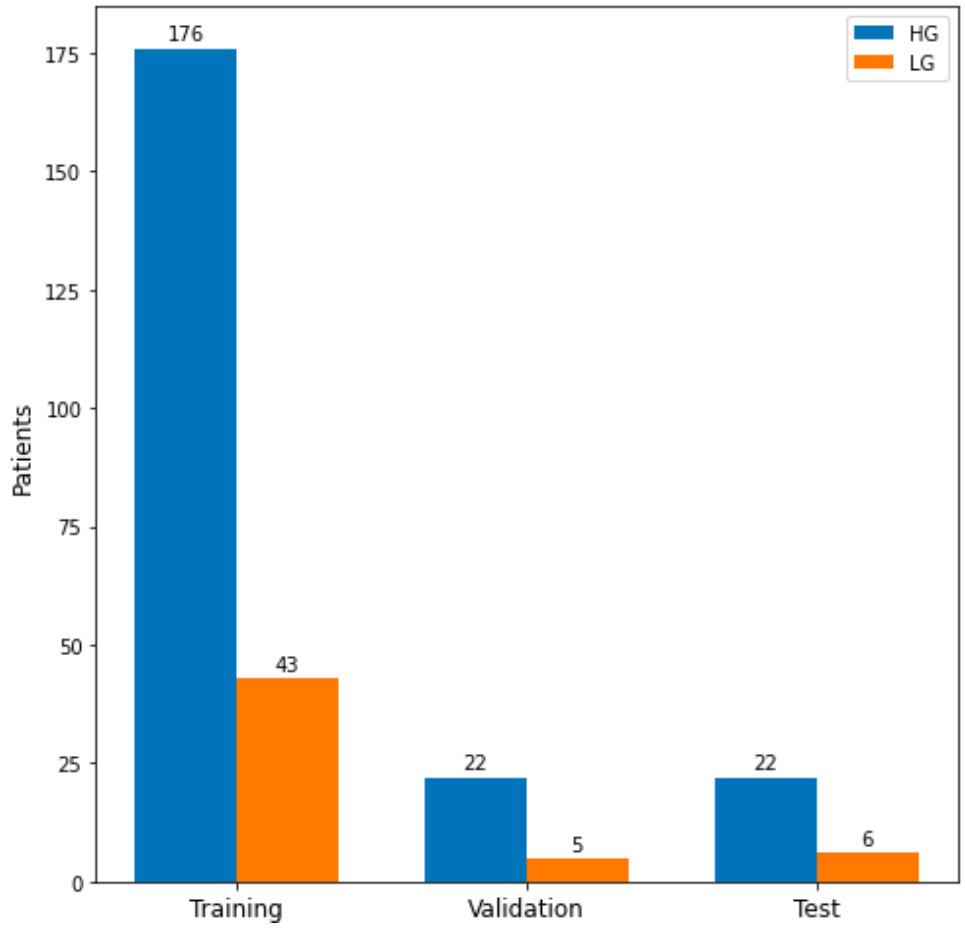


Figure 3.3. Data distribution by set and tumor grade, after the split of BRATS2015 in Training (80%), Validation (10%) and Test (10%) using stratified sampling.

BRATS2015 contains basically images and few additional information about the patient: a unique identifier, not much useful, since data are anonymized (beside the fact we used it to find the correct ground truth folder of each volume), and whether the subject is a low grade or a high grade.

3.3.2 Images

The images are the most interesting part of the dataset: in total there are 1370 volumes and since, from each volume of the brain, it's possible to extract 155 2D axial slices, with dimension 240x240 (an axial slice is one of the possible perspectives from

which is possible to represent the brain in two dimensions. The other two principal planes are the coronal and sagittal ones.), the number of available images is 212.350. The images are grayscale, so, instead of multiple color channels, there is just one channel that carries information and each value of a pixel represents only an amount of light.

Per each patient there are 5 volumes: 4 of these contain the different MRI sequences of the dataset (T_1, T_2, T_{1c}, T_{2flair}) while the last volume corresponds to the segmented area of the tumor (in BRATS2015 this volume is called *Ground Truth*).

The modalities contained in the dataset are some of the most commonly acquired MRI sequences: we already discussed about T_2-weighted (T_2) and T_2-fluid-attenuant inversion recovery (T_{2flair}). In addition to these two, we have T_1-weighted (T_1) and T_1-with-contrast-enhanced (T_{1c}). These four scans provide both redundant and complementary information about the imaged tissue and each one of this can give important and unique insights about the interested zone, in our case the brain.

T_1 and T_{2flair}, for example, give meaningful information of the edema region of tumor in case of glioblastoma. T_{1c} on the other hand can become become very useful when a contrast agent can be used with the patient, since it defines a clear demarcation of enhancing region around the tumor that can be used as an indicator to assess growth/shrinkage. T_2 is used instead to detect hyperintensities that can lead to the diagnosis of vascular dementia [1].

The 5th type of image given in this dataset is the one representing the tumor area (in BRATS2015 every subject has a brain tumor) that was used in this work to evaluate the results obtained by the trained models, as it happens in the segmentation task.

In segmentation though, these ground truths, made typically by one or more human experts, are used to quantify how good an automated segmentation is with respect to true tumor area.

The difference here is that we used the Ground Truth mainly as a mask applied to the synthesized images in order to discard the area of the brain not related to the tumor and then to compute the metrics and measure the quality of the generation only in that specific area, that is the most interesting and important one for the final diagnosis.

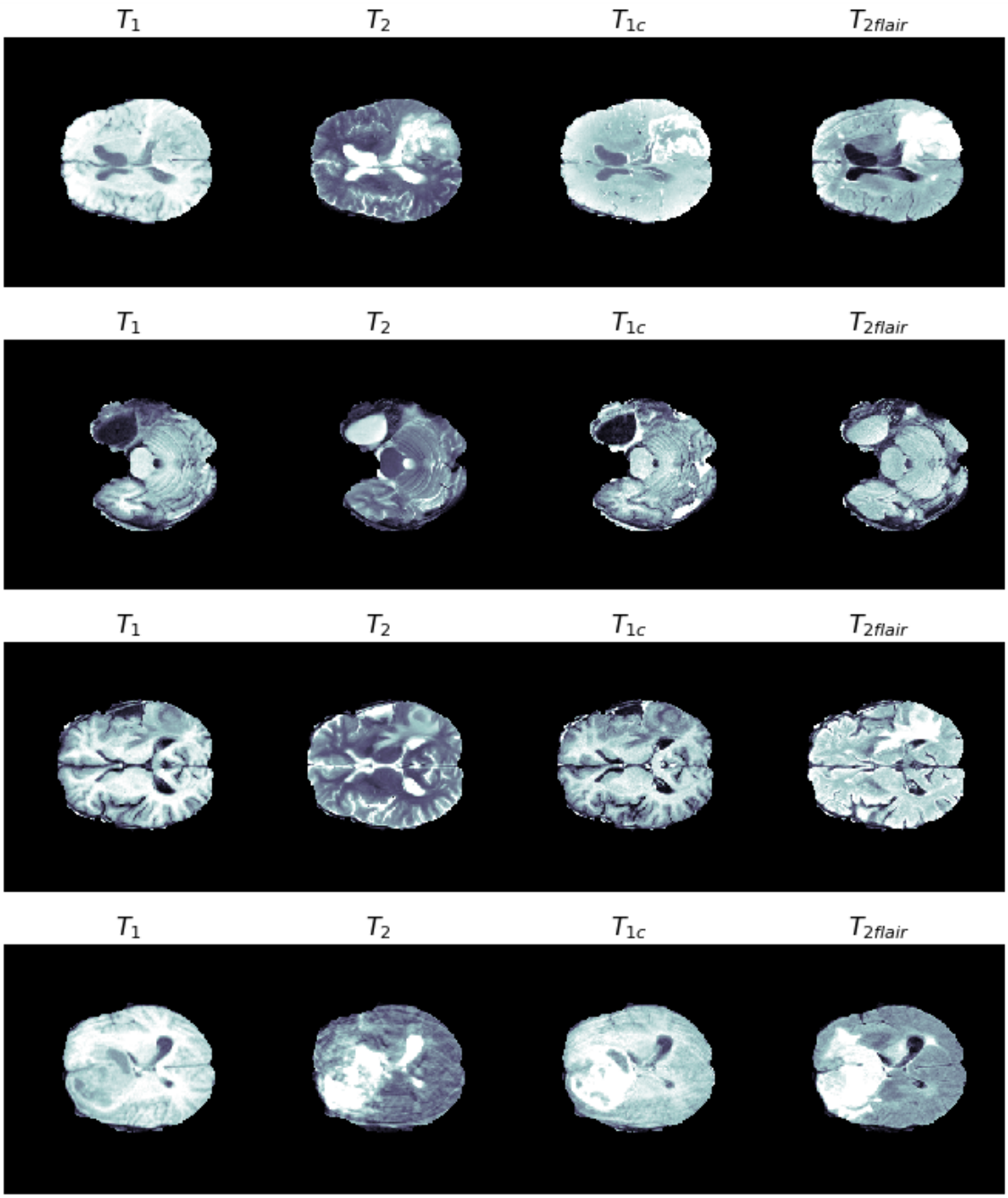


Figure 3.4. Brain MRI sequences - T_1, T_2, T_{1c}, T_{2flair} - from four different patients.

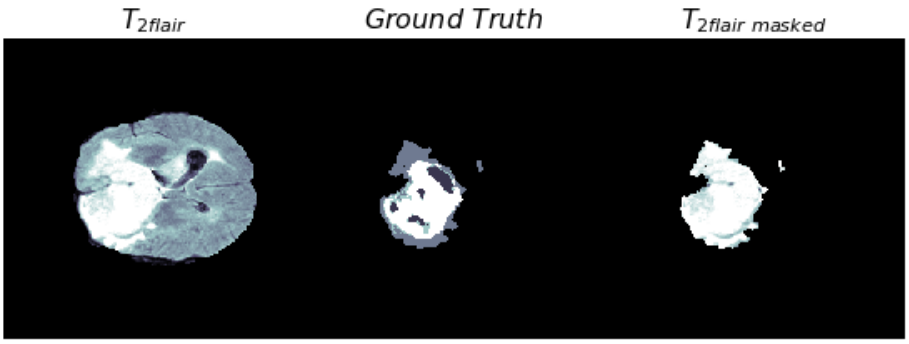


Figure 3.5. $T_{2flair\ masked}$: tumor area obtained using the Ground Truth as mask over T_{2flair}.

3.4 Preprocessing

As in almost every deep learning pipeline, some preprocessing of the dataset was required: the details of the transformations applied to prepare the data in the proper manner are presented below.

3.4.1 Cropping

All the images in the dataset were first center cropped: the outer parts of each volume were removed while the central region was retained along each dimension.

This first preprocessing step reduced the dimensions of each volume from [240, 240, 155] to [180, 180, 128] but it's important to notice that no useful information was removed but black voxels, so only the outer parts of the MRI with zero intensity value. This cropping brought also the advantage of having less heavy data in terms of size: removing useless information (as the black pixels around the brain) from a large dataset means, when it comes to read the data, less time to load the images in memory and obviously less memory needed to cache everything.

We observed that not all the 155 slices are equally relevant to us: some of the most external ones in the volume are totally black images while some contain just few pixels with intensity value different from zero.

These images of course are meaningless for us since they contain neither enough information about the tumor nor significant brain structure that could be helpful to the GAN training: because of this we reduced the number of slices from 155 to 128. Figure 3.6 shows an example of the first 48, out of 128, T_2 slices extracted from the volume of a subject, after the application of central cropping.

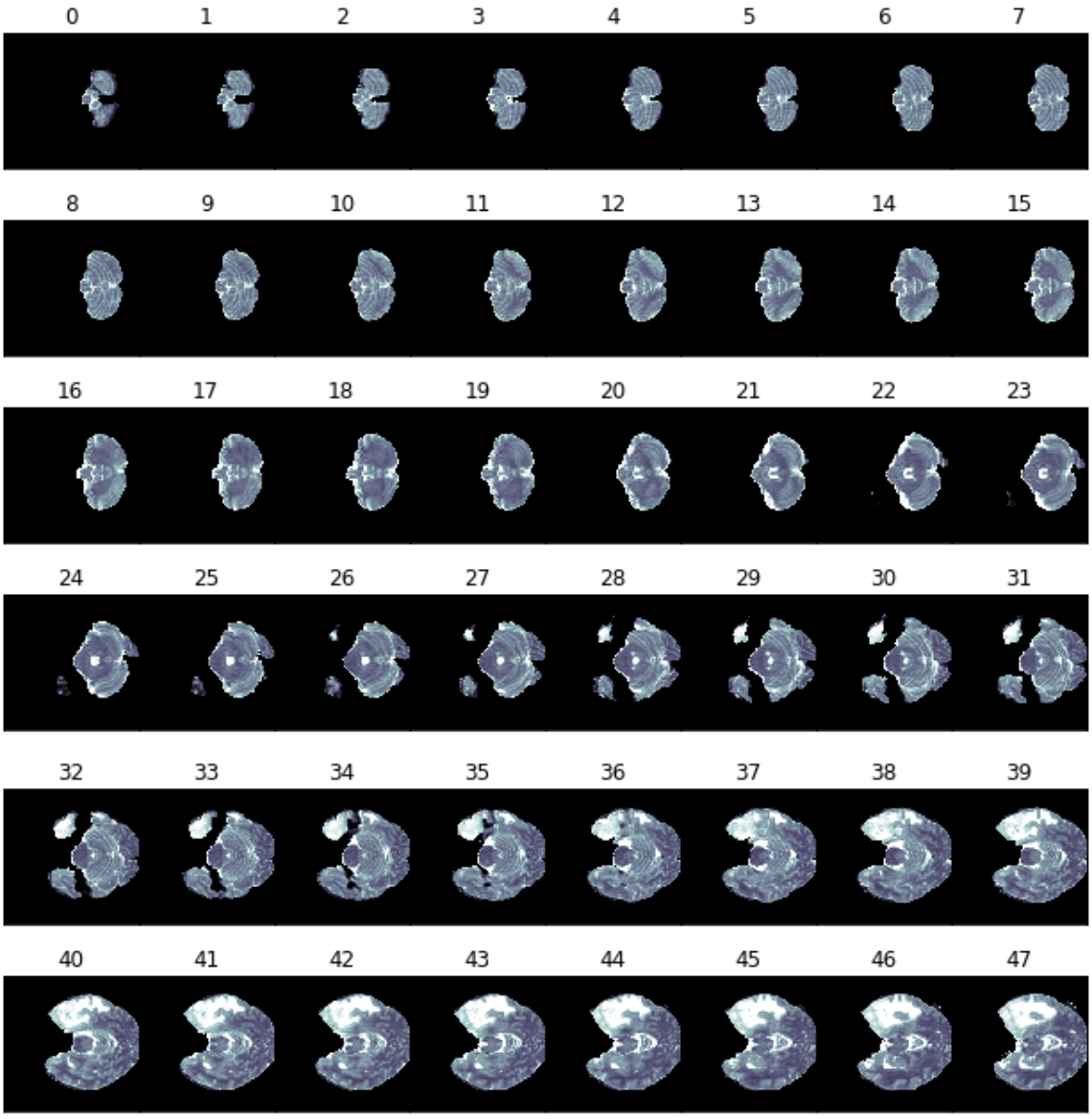


Figure 3.6. An example of the first 48 T_2 slices extracted from a volume, after cropping it to a width of 180, height of 180 and depth of 128 (num. of axial slices).

3.4.2 Normalization

The motivation to normalize the images before feeding them into our models comes from the fact that in many machine algorithms it is required to have the same dynamic range, so the difference between the maximum pixel value and the minimum pixel value, per each image: in this way, by converting an input image to a range of pixel values that is more familiar and normal to the sense, it's possible to avoid undesired effects.

In our case we used a type of rescaling known as min-max scaling or min-max normalization where the transformation, since we're dealing with gray scale images, interested only one channel.

Min-max is the simplest normalization method and allows to bring the difference between the max value and the min value of an image to be in the range [a, b] using the following formula:

$$x' = a + \frac{(x - \min(x))(b - a)}{\max(x) - \min(x)} \tag{3.1}$$

In our case we wanted to bring all the images to lie within the range of 0 to 1 by using the Formula 3.1 that with a = 0 and b = 1 becomes:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3.2}$$

where, given a volume of a subject, x is the original intensity value and x' is the new normalized voxel value. The min and max values were set to the values corresponding, respectively, to the 2nd and the 98th percentiles of the voxels intensities of the whole volume, to avoid the risk of using outliers in the normalization process: in fact we observed that in the dataset there are unusual high-intensity values due to pathologies and without this shrewdness these high values would have squashed the pixel range to always lie very close to zero.

Table 3.1 and 3.2 show how much important was to apply this processing step before actually starting to work with the data. Three volumes are used as example: the first two rows correspond to the same modality, T_2, but different subjects while the 2nd and the 3rd row show minimum value, maximum value, 2nd percentile and 9th percentile for the T_2 and T_{2flair} of the same patient. The voxels values shown can't be really used without applying some scaling to the intensity channel neither to compare images from the same modality nor to evaluate volumes from the same patient.

All the 128 slices belonging to the same person are normalized with respect to

Patient	Modality	Min value	Max value	2nd Percentile	98th Percentile
0011_1	T_2	0.0	1344.0	0.0	579.0
117_001	T_2	0.0	2648.0	0.0	1013.0
117_001	T_{2flair}	0.0	720.0	0.0	389.0

Table 3.1. Values from different volumes before normalization

Patient	Modality	Min value	Max value	2nd Percentile	98th Percentile
0011_1	T_2	0.0	1.0	0.0	1.0
117_001	T_2	0.0	1.0	0.0	1.0
117_001	T_{2flair}	0.0	1.0	0.0	1.0

Table 3.2. Values from different volumes after normalization

fixed percentile values: for example all the slices with 117_001 as patient id and T_{2flair} as type of modality will be rescaled by using Formula 3.2 where 389.0 (the 98th percentile) is the max value (instead of 720.0 that is an outlier value) and 0.0 (2nd percentile) is the min value. Results of the normalization are shown in Table 3.2.

This transformation step in the preprocessing phase was necessary because not all the images had the same dynamic range since BRATS2015 contains scans from multiple sources: the MRI volumes were obtained in different hospitals and with different acquisition settings (even the sequences of the same subject). By rescaling the whole dataset we were able to obtain more homogeneity among images, ranging now between 0 and 1.

3.5 Summary

In this chapter we presented some details about Magnetic Resonance Imaging, describing the problem of missing modalities, and the reason why we focused our attention on brain tumor MRI scans.

We gave then an overview about the brain MRI dataset used in this work, BRATS2015, discussing how it is organized and which modalities it contains, showing also some figures to give an idea to the reader of the appearance of the different scans used to generate the missing modalities.

At the end of the chapter we described the preprocessing transformation useful to prepare the data before being fed into the models, whose architecture will be discussed in Chapter 4.

Chapter 4

System Design and Overview

The purpose of this chapter is to give a detailed description of how we optimized the input pipeline (Section 4.1) of our models and how we built the architecture of the networks used (Section 4.2) in this work. In Section 4.3 we present details from the training algorithm and which are the losses used in the update step, while Section 4.4 illustrates the evaluation metrics the we used in order to have a quantitative measure of how good were performing the trained networks.

4.1 Input Pipeline

After preprocessing the dataset with the transformations described in Section 3.4 (cropping and normalization), we proceeded to store the data as a *TFRecord*.

TFRecord, provided with the free and open-source software library of *Tensorflow*, is a format that allows large amount of data to be read in an efficient way [32] by serializing it and storing it as a sequence of binary records of varying sizes: GPUs, together with an efficient input pipeline (that starts from choosing the right data format), allow to achieve peak performance and to reduce computational times.

In order to build an highly performant input pipeline for the models we applied some important operations to our TFRecord data including a preprocessing step on the fly while loading data with the *Data API*:

- **Caching** the dataset in memory to save operations, such as file opening and data reading, from being executed during each epoch. It was crucial to apply this step before shuffling, batching and prefetching: data is read only once but is shuffled (next step) every time in a different way, with the next batch still being prepared in advance [19, p. 915].

- **Shuffling** the training set (with buffer size set to 128) since *Gradient Descent* works best when the instances are independent and identically distributed.

- We then performed **Mapping** to decode efficiently the binary information and store it into tensors. Inside this step, another processing transformation was applied to the data: a constant (zero - value) padding was added to images to reshape them as [256, 256, 128], i.e., a shape compatible with the input generator of the GANs implemented (see Section 4.2 for more details).

- **Batching** plays a major role in the training of deep learning models and because of this it was crucial to choose the right batch size based on the GPU available and on the neural networks that we were going to train.

  Choosing the right batch size is important because it determines how many images are used to train the model before updating all the weights and biases. In our case we identify, after performing various experiments varying the batch size from 1 to 256, that 32 was the right number of images to use, at each iteration of the training algorithm, as input for all the models implemented.

Batch size	Batches/epoch	Train time/epoch (s)
1	1024	≈ 500
4	256	≈ 130
8	128	≈ 75
16	16	≈ 40
32	**32**	$\approx$ **30**
64	16	≈ 25
128	8	≈ 23
256	4	GPU out of memory

Table 4.1. Train time/epoch of pix2pix using a reduced set of 1024 images.

Table 3.2 shows some of the experiments we performed to determine the proper batch size, evaluating the training time taken by pix2pix (Subsection 4.2.1) on GPU with a reduced dataset of 1024 images, varying the size of the batch and consequently the number of batches.

We noticed that a batch size equal to 256 couldn't fit the GPU RAM provided by Google Colaboratory: the processors were running out of memory giving an error of resource exhausted. Furthermore, even though the training time was slower with respect to the times obtained with batch size of 64 and 128, we observed that 32 reached, qualitatively, more accurate results in the generated samples and [19, p. 681] explains the reason: in practice, large batch sizes often

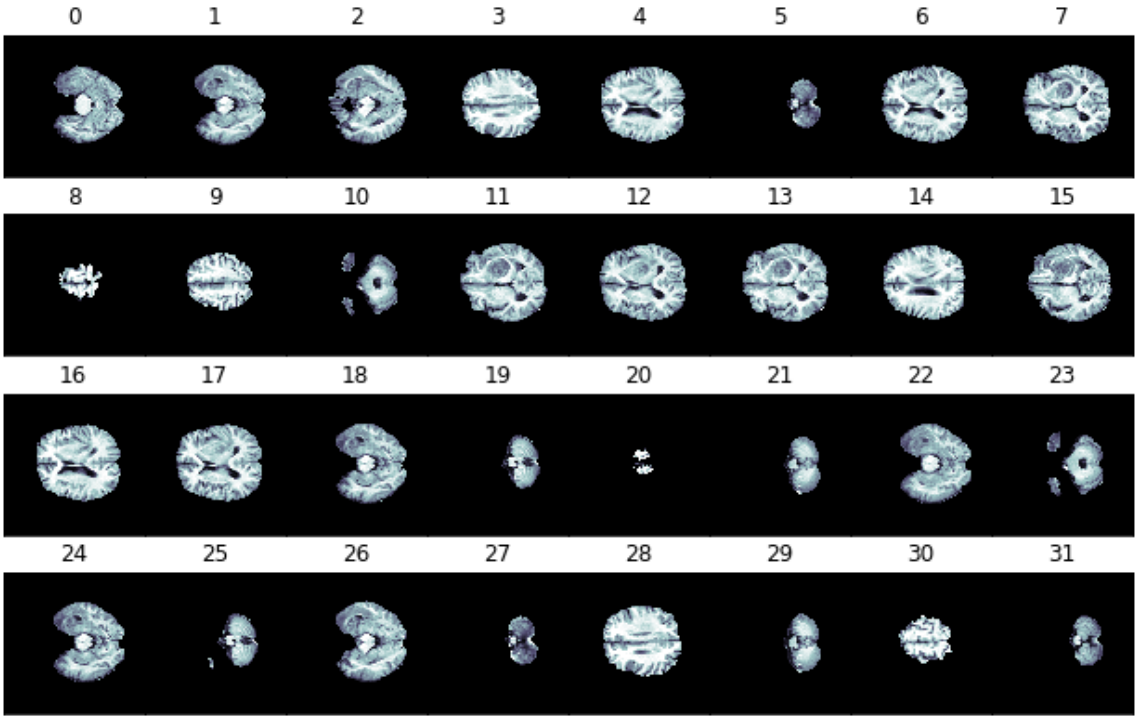


Figure 4.1. Example of single channel batch from the training set, with 32 T_1 slices cropped, normalized, shuffled and padded, with dimensions 256x256, ready to be fed to a single input model.

can lead to training instabilities and so the models may not generalize as well as a model trained with smaller batch size.

- Finally we applied a **Prefetching** step to have the preprocessing overlapping with the training execution: while the model is executing training step *s*, the input pipeline is saving time by reading data that will be used in the next training step *s+1*.

4.2 Neural Networks Architecture

In this work we developed two Generative Adversarial Networks: a multi-input version of pix2pix [6], that we called MI-pix2pix, and a modified version of the MM-GAN proposed by Sharma et al. in [1]. We also implemented pix2pix that was used mainly as a baseline to compare the performances reached by our multi-modal GANs, to see if they were producing better results, both in a qualitatively and quantitatively way.

In this section we discuss in detail the architectures behind these three GANs, showing also the differences between the model proposed by Sharma et al. and our modified version.

4.2.1 Pix2pix

Pix2pix is a single input - single output GAN, proposed by Isola et al. in [6] as a solution to the problem of image-to-image translation, extensively discussed in Subsection 2.2.5. This Generative Adversarial Network adopts a modified U-Net architecture [23] as generator and a PatchGAN [24] as discriminative model that tries to distinguish between the true samples coming from the dataset and the synthesized images of the generator.

Generator

In Subsection 2.2.5 we discussed about the strength of U-Net as generative network: skip connections between mirrored layers that represent a solution to the problem of losing information through the bottleneck layer. These connections are realized linking the i layer to the n-i layer (where n is the total number of layers) that receives as input a concatenation between the output of the i layer and the information processed by the *n-(i+1)* layer.

The generator architecture, with the U-shape from which U-Net takes its name, is illustrated in Figure 4.2. The input of the model is a tensor with size 32x256x256x1, where the first one indicates the batch size, so how many images are used per time for a single update.

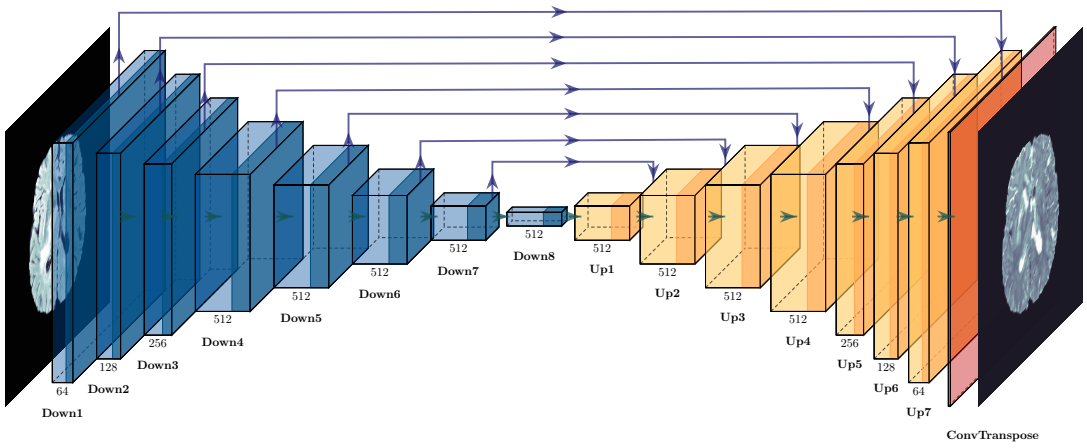


Figure 4.2. Generator architecture from pix2pix [6].

In Section 4.1 we explained that it was necessary to add a padding to our images because otherwise they wouldn't have been compatible to the input of the images: the problem with U-Net comes from the nature of its mirrored architecture and more in particular from its downsampling and upsampling layers that allow to have only input images with dimensions that are power of 2. We choose then to pad our 180x180

images in order to obtain a width and height of 256, instead of applying a resize/crop to 128: transformation that wouldn't have preserved all the information in the images.

The input fed to the model goes first through the contracting path: a typical convolutional neural network (Subsection 2.2.1) composed by the repeated application of one the two main building blocks of the network, the **downsample block**, composed as follows:

- **Convolution**, that halves the dimensions of the input by convolving it through several filters with a fixed kernel size equal to 4 and stride of 2 for all the spatial dimensions.
- **Batch Normalization** in every block (with the exception of the first one). This operations zero-centers and normalizes each input, trying to overcome to the problem of the vanishing gradient [19, p. 728].
- **Leaky ReLU**, an activation function similar to ReLU but with a small slope for negative values useful to avoid that some neurons "die" during training (meaning that they stop outputting anything other than 0) [19, p. 720].

The repeated downsampling reduces the spatial information while increases the feature dimension, until the input arrives at the bottleneck where the output shape of the last downsampling block (*Down7*) is [1x1x512].

The building block that characterizes the expanding path of the network is instead the **upsample block**, composed by four layers:

- **Transposed convolution** that goes in the opposite direction of what is done by the Convolution, by enlarging the spatial information: image is first stretched with the insertion of empty rows and then a regular convolution is computed. A kernel size of 4 and stride of 2 was set in this layer.
- **Batch Normalization**
- **Dropout**, useful for its regularization effect against some possible overfitting, it was applied in the first three upsampling blocks.
- **ReLU** that has the advantage to not saturate for positive values (and it is fast to compute) [19, p. 720]

The last layer in the architecture of the generative part of [6] is another transposed convolution that prepares the output of the model with tanh (hyperbolic tangent) as

activation function, that returns values in a range between -1 and +1. The output of the network is a batch of images, 32 in our case, with dimension 256x256x1.

Discriminator

The PatchGAN [24] used to distinguish true images from fake images is similar to the well-known Convolutional Neural Network architecture (Section 2.2.1), with the difference that the discriminator of pix2pix doesn't discriminate the whole image but small NxN patches of the input image: in this way the network can focus on capturing high frequencies details and assign to each patch a fake or true label.

This means that the output of the discriminator we used, in pix2pix and in the other GANs we experimented with, isn't a 1x1 value but a 30x30 matrix, where each value contains the result of the discrimination over a 70x70 portion of the input image (the authors that developed [6] suggest to use a patch of this size to obtain the best results from the GAN).

Because of this, the network used is called 70x70 PatchGAN and takes two inputs, as illustrated in Figure 4.3, each with shape [32, 256, 256, 1]: the first one is an input image while the second one is the target that can be an image coming from the dataset or a sample generated by the GAN.

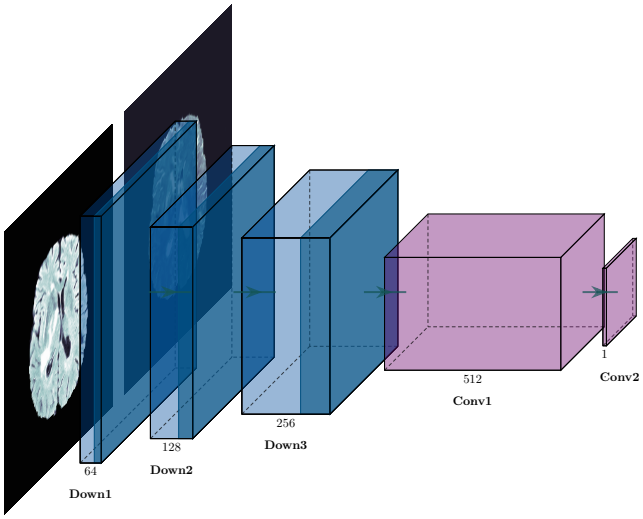


Figure 4.3. Discriminator architecture from pix2pix [6].

The discriminator architecture, differently from the one of the generator, has only a contracting path and is composed by three downsample blocks that allow to reduce the spatial dimensions of the image while increasing the feature dimension. Each one of these blocks is characterized by a different number of filters (64 the first block, 128 the second one and 256 the last one) and they are defined with the same layers that

compose a downsample block in the generator (so batch normalization applied only in *Down2* and *Down3*.

These blocks are then followed by several layers:

- **Zero Padding**, that doesn't increment the number of trainable parameters, because it simply adds zeros at the top, left, right and bottom of an image.
- **Convolution** with 512 filters, kernel size equal to 4 and stride of 1.
- **Batch Normalization**
- **Leaky ReLU**
- **Zero Padding**
- Final **convolution** with 1 filter, again with kernels of 4x4 and stride set to 1.

The description of the training algorithm used by pix2pix can be found in Subsection 4.3.2 while the implementation details are discussed in Subsection 4.3.3.

4.2.2 MI-pix2pix

MI-pix2pix is the second Generative Adversarial Network that we included in our experiments: it is a modified version of pix2pix [6] adapted to a multi-input scenario in order to compare and evaluate the differences between an unimodal approach and its variant that takes multiple modalities of available information as input.

Generator

The layers that compose the generator architecture (Figure 4.4) are the same as the ones that define the pix2pix model with the only difference in the input shape, that is set to [32, 256, 256, 3], while the output shape remains equal to [32, 256, 256, 1]. For example a MI-pix2pix that has to synthesize the missing T_2 sequence of a subject will use the information from the other three modalities available: T_1, T_{1c}, T_{flair} that are the inputs respectively of the first, second and third input channel.

Discriminator

Also for the Discriminator we maintained the same architecture (Figure 4.5) used in the unimodal version of pix2pix, but, since we are in a multi-modal setting, while the target has still a shape of [32, 256, 256, 1], the second input is a concatenation of the three modalities ([32, 256, 256, 3]) used to feed also the Generator.

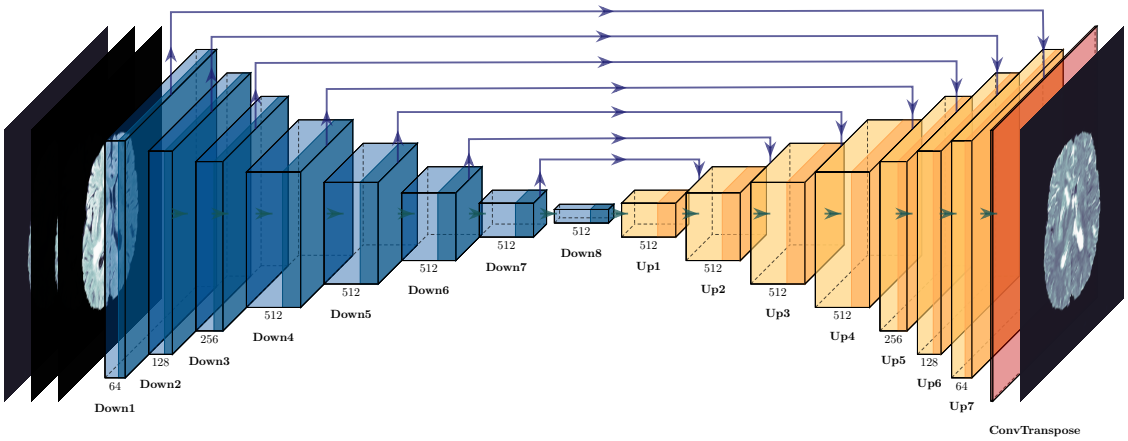


Figure 4.4. Generator architecture from MI-pix2pix.

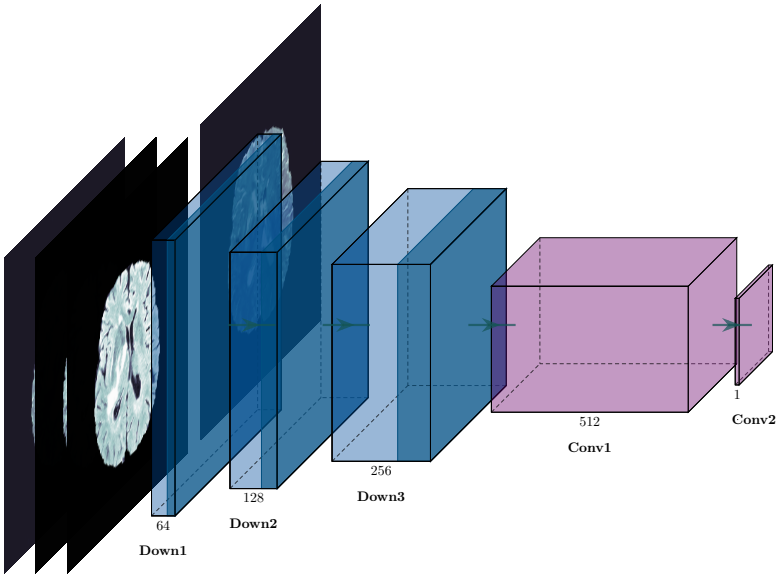


Figure 4.5. Discriminator architecture from MI-pix2pix.

MI-pix2pix was very useful to understand whether a multi-modal approach can exploit the information coming from different MRI sequences in a better way than what is done by pix2pix, since the two GANs share the same architecture and the same layers but receive a different number of images. The results of the comparison are presented in Chapter 5.

As for pix2pix, description of the training used can be found in Subsection 4.3.2 while the implementation details are discussed in Subsection 4.3.3.

4.2.3 MI-GAN

The third Generative Adversarial Network that we implemented is a multi-modal approach inspired by the work of Anmol Sharma and Ghassan Hamarhneh that in 2019 presented a multi-input multi-output GAN, called MM-GAN, that seems to be, to the best of their knowledge, the first method capable of synthesizing multiple missing sequences using a combination of various input sequences (many-to-many) [1].

Since the core of this work is on the generation of a single-output image, using different configurations and architecture of GANs, we focused our attention on a modified version multi-input single-output (many-to-one) of MM-GAN, called MI-GAN and implemented also by Sharma and Hamarhneh.

The focus of [1] was on the many-to-many approach though, and as a consequence many details about the MI-GAN model weren't given: because of this, our model was inspired by the MM-GAN but at the same time it presents some variations that we applied, both to the architecture and to the loss used in the training algorithm (Subsection 4.3.2), in order to obtain a more stable training and satisfiable results in the generated samples.

The layers that define the architecture of MI-GAN are almost the same proposed in [1] for the MM-GAN: the only variation applied was the replacement of every layer of Instance Normalization with a Batch Normalization, used also in pix2pix (Subsection 4.2.1) and in MI-pix2pix (Subsection 4.2.2), after observing through various experiments that normalizing the activations of each channel across the whole batch was more effective, in terms of quality in the generated samples, instead of computing the mean/standard deviation and normalizing across each channel in each training image.

Generator

The generator is a modified U-Net with concatenated skip connections and the typical U-shape architecture (Figure 4.6), characterized by two known building blocks: one that downsamples its input and one that, through a transposed convolution, performs an upsampling.

The **downsample block** is composed in a similar way to the one used in the previous model, with the addition of a dropout layer at the end of the block.

- **Convolution** with kernel size equal to 4 and stride of 2.
- **Batch Normalization** in every block beside the first one.
- **Leaky ReLU**

- **Dropout** with the rate defining the fractions of the input dropped to 0, at each update during training time, set to 0.5.

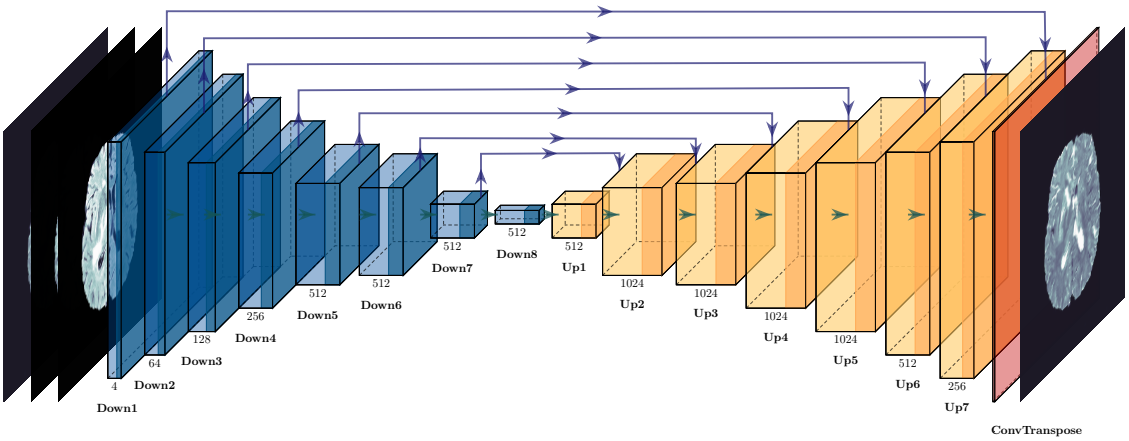


Figure 4.6. Generator architecture of MI-GAN.

The **upsample block** instead, following the MM-GAN architecture, has the same layers seen in the pix2pix architecture with the difference that the dropout layer is applied after the ReLU activation function:

- **Transposed Convolution**
- **Batch Normalization**
- **ReLU**
- **Dropout** (with 0.5 rate) applied to the first three upsample blocks.

At the end of the network, after the last upsample block, a transposed convolution with tanh as activation function is used to reduce the feature dimension and to enlarge the spatial dimensions of the image from 128x128 to the final shape of 256x256.

Regarding the problem of how many input and output channels use in the generator, we performed two experiments with the architecture above defined: in the first one, following [1] and adapting the MM-GAN network from a many-to-many to a many-to-one scenario, we trained a generator to produce 4 different modalities (but only the synthesized image of the missing sequence was used to compute the loss, while the other three were discarded) receiving as input a concatenation of 4 images. These images given as input, in the case for example of T_1 missing, were the three remaining modalities $\{T_2, T_{1c}, T_{2flair}\}$ and a zero-value image through the channel of the missing

sequence (so the channel of T_1 was obscured). The second approach we tested was about using a generator with the same input and output shape already used in MI-pix2pix: a single image with shape [32, 256, 256, 1] generated by a model that receives three 32x256x256 different modalities.

The results obtained led us to discard the first solution, probably more functional in a many-to-many scenario, and to choose the second approach that outperformed the other one both in a qualitatively (human perception of quality from the generated samples) and quantitatively (scores from the similarity/error metrics applied to the synthesized images) way.

Discriminator

The experiments we did in order to find the best suitable network involved also the Discriminator: with the generator synthesizing 4 images, we implemented the model discussed in [1] that takes two inputs X_t and X_i and produces one output D(X_t, X_i), each one of these with 4 channels, one per modality: assuming the discrimination of a T_1' synthesized scan, the inputs were X_t: $\{T_1, T_2, T_{1c}, T_{2flair}\}$ and X_i: $\{T_1', T_2, T_{1c}, T_{2flair}\}$.

In second experiment, the one with the generator producing only one image, the two discriminator inputs, input and target, had respectively shape [32, 256, 256, 3] and [32, 256, 256, 1] with an output with dimensions equal to 32x15x15x1 (in the discrimination of a T_1' scan, the inputs were $\{T_2, T_{1c}, T_{2flair}\}$ and T_1'). As said before, we choose to use the second solution, over the first one.

The discriminator is a PatchGAN [24] composed by four downsampling steps called **discriminator blocks**, similar to the ones used by the MM-GAN [1]. Each block contains, in order, the following layers:

- **Convolution**, with a number of filters ranging from 64 (first block) to 512 (last block).
- **Batch Normalization** applied in the second, third and fourth block.
- **Leaky ReLU**

At the end of these sequence of blocks, two more layers are applied (Figure 4.7):

- **Zero Padding**
- Final **Convolution** with kernel size to 4 and strides set to 1.

The description of the training algorithm as well as the loss formulation used by MI-GAN is in Section 4.3, while the implementation details are discussed in Subsection 4.3.3.

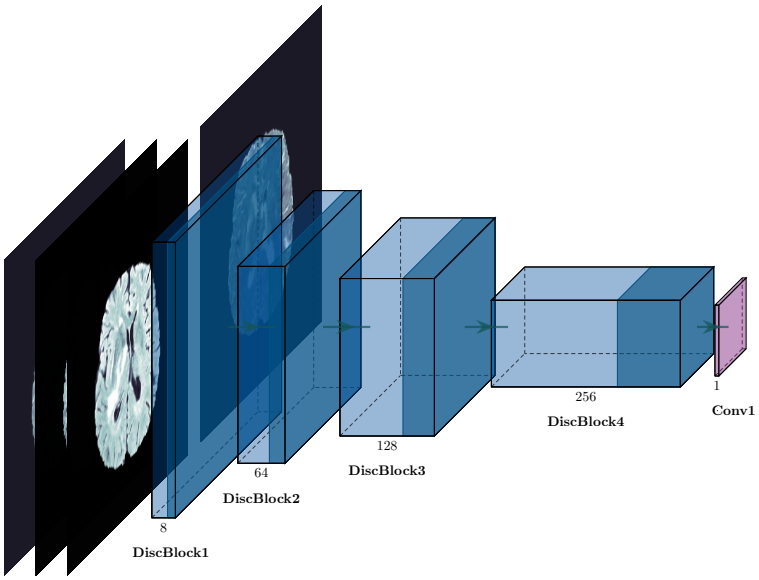


Figure 4.7. Discriminator architecture of MI-GAN.

4.3 Training

In a Generative Adversarial Network, generator and discriminator are trained simultaneously using *Backpropagation*, with D that learns to recognize the true images from the fake ones and G that, based on the feedback received by the first neural network, learns to produce more realistic images in order to be able to fool D. In this adversarial setting, the competing behaviour of one against each other is defined by two different loss functions.

4.3.1 Loss Formulation

The generator loss and discriminator loss used with MI-GAN (4.2.3) are the ones proposed in [6] and defined as follows:

$$\begin{aligned} L_G &\leftarrow \lambda \mathcal{L}_1(G(x), y) + (1-\lambda)\mathcal{L}_2(D(x, G(x)), L_{ar}) \\ L_D &\leftarrow \mathcal{L}_2(D(x, y), L_{ar}) + \mathcal{L}_2(D(x, G(x)), L_r) \end{aligned} \tag{4.1}$$

where x, the input, is a concatenation of three sequences while $G(x)$ represents the prediction generated by the GAN. L_{ar} is a dummy ground truth tensor that is

used by the generator loss to fool the discriminator by setting all the entries to one, in such a way that G is encouraged to generate modalities that can fool D.

$\mathcal{L}_2$ is the L2 norm (also known as mean squared error) while $\mathcal{L}_1$ instead is the mean absolute error (L1 norm) that is useful to the generated image because it allows to become structurally similar to the target image. This term was chosen as reconstruction loss term because of its ability to prevent too much blurring, as compared to using a L2 loss [1]. L_r is instead a tensor with its entries equal to zero, useful for D to distinguish the fake samples (associated to 0, by using L_r) from the true samples (associated to 1 through L_{ar}).

Pix2pix (4.2.1) and MI-pix2pix (4.2.2) share the same objective function (Eqn. 2.4), since they have an identical architecture and the only difference between the models is in the number of inputs that their generator can receive.

In the implementation of these two networks, we applied the same losses that are used in [33], similar to the ones adopted in the MI-GAN training algorithm: here, a binary crossentropy is used instead of the L2 term.

The L_D of pix2pix and MI-pix2pix is computed as the sum between the binary crossentropy of $(D(x, y), L_{ar})$ and the one of $D((x, y'), L_r)$ where y' is the generated image. L_G, on the other hand, contains the reconstruction term between y and y', multiplied by an hyper parameter lambda and summed to the binary crossentropy of $D((x, y'), L_r)$.

4.3.2 Training Algorithm

Due to some implementation needs and to be able to define our own metrics (Section 4.4 for details) and losses, we wrote a custom train step (where is contained the update of the weights) as well as evaluation loops instead of using the built-in training optimized by *Tensorflow*. The two approaches work strictly in the same way across every kind of *Keras* model, but implementing everything from scratch allowed us to have a lower-level training and full control on the evaluation step, as well as the possibility to being able to retrieve all the gradients of the trainable weights of a layer with respect to the loss.

For implementing the training algorithm we followed the algorithm applied in [33], which uses *Adam* [34], an adaptive learning rate method, as optimizer for both the networks.

During the training of our models, we didn't make use of the regularization technique called *Early Stopping*, used to prevent overfitting in many DL methods.

This technique is based on the idea of stopping the training procedure before the model starts to overfit, by looking at the performances on the validation set, in particular at the validation loss: when its value, after an initial improvement of the learner's ability to generate/predict/classify, starts to increase, it means it is probably the right moment to stop the training because the models is incurring into a larger generalization error. In our case though, we observed that, often, after an abrupt increase of one of the two losses, the networks after few epochs were returning to a stable training level, without the need of killing the procedure and restarting again from the beginning.

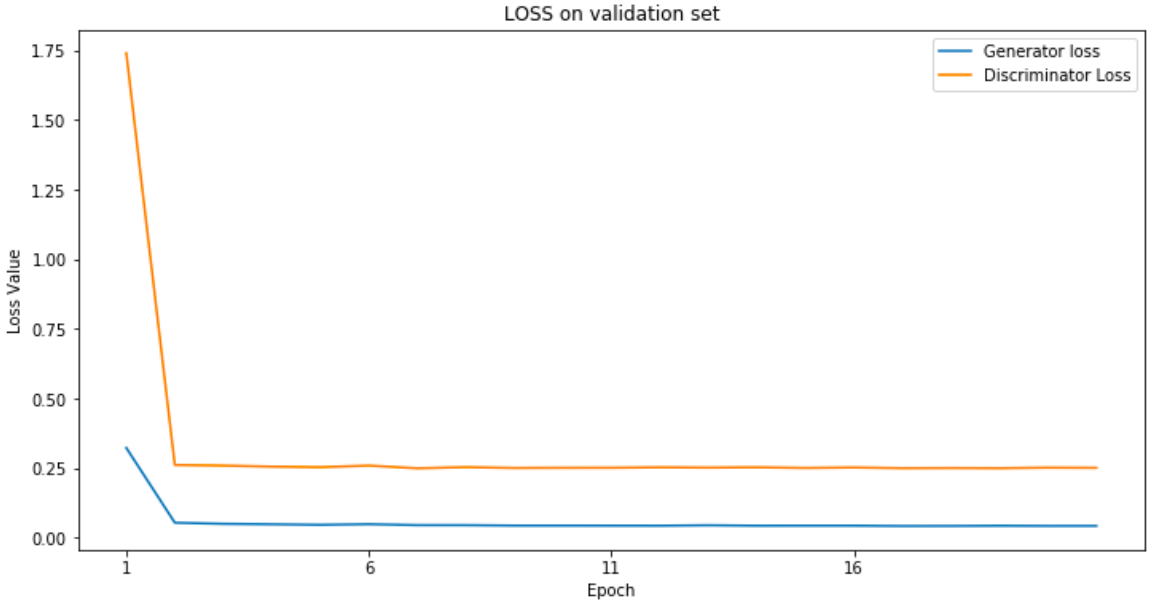


Figure 4.8. Typical behaviour of the validation losses through the epochs of MI-GAN training.

Furthermore, our Generative Adversarial Networks continued to improve the overall quality of their generated image even if their losses values, the indicators used to understand when we should stop, had been stable for 10, 20, even 30 epochs: GANs took long time to train and long time to show some progress in the synthesized samples, that's why it was important to wait for a while before killing the training. The only exception is represented by the case in which the loss went rapidly to 0, sign that something wasn't working in the proper way and that the network had incurred in a failure to converge, that can happen for example when the discriminator loss drops to 0 because it is learning more rapidly with respect to the generator.

Figure 4.9 illustrates the differences between the original T_{1c} slice and its prediction, from MI-GAN, after 5 and 20 epochs of training: it shows how much a GAN can improve the quality of the generated images, adding low-level details and reducing the overall blurriness, even when the losses, both in the generator and in the discriminator,

are not improving over time (Figure 4.8): proof that, during training, a visual understanding is much more meaningful than just looking at the values of the losses.

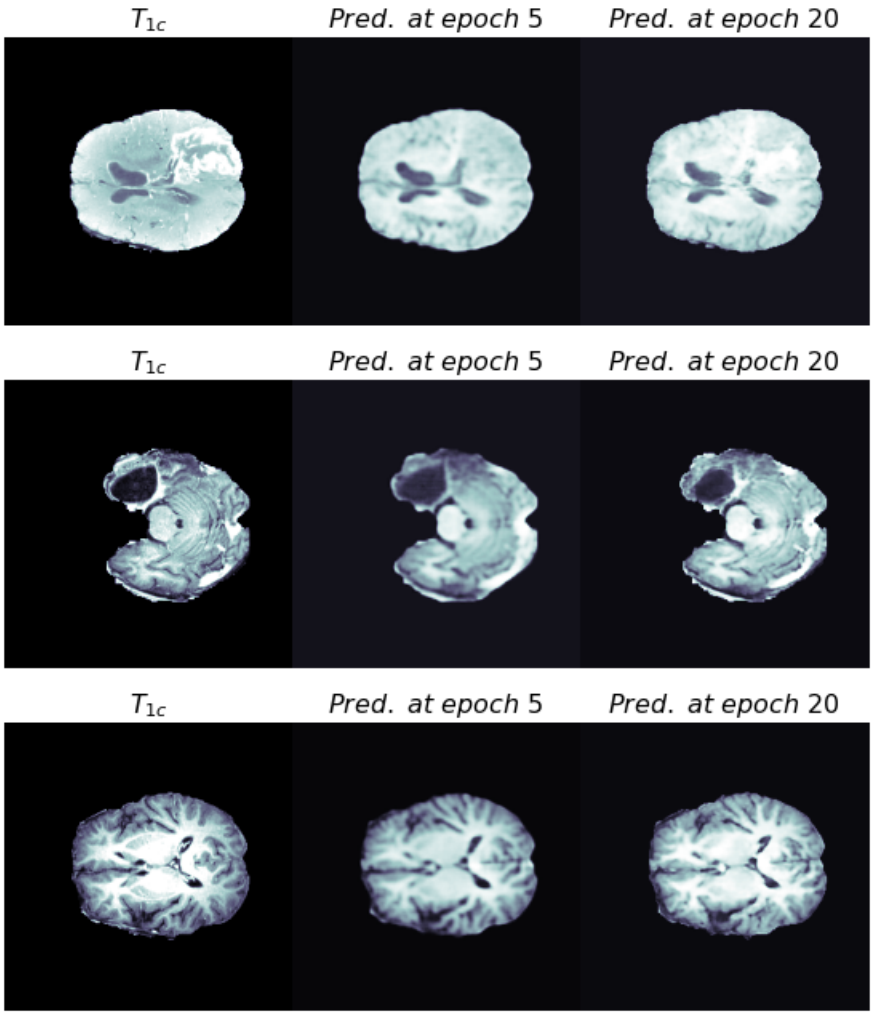


Figure 4.9. Prediction of T_{1c} slices on the validation set, using MI-GAN.

4.3.3 Implementation Details

The hyperparameters used are the learning rate α set to 0.0002 and the exponential decay rate for the 1st moment estimates β_1 equal to 0.5 while the one for the 2nd moment estimates (β_2) was left to the default value of 0.999.

λ has a value of 100 in the generator loss of pix2pix and MI-pix2pix, while is set to 0.9 in the loss of the MI-GAN generator. The MI-GAN discriminator loss instead is multiplied by 0.5, choice motivated by the need to slow down the rate at which D learns compared to G.

Furthermore, the transposed convolution layers in the generators architectures (whose shapes are summarized in Table 4.2) as well as the convolutions used by both generators and discriminators (Table 4.3) have a variable number of filters with dimension 4x4 and with kernel weights initialized using a normal distribution with 0 mean and standard deviation equal to 0.05.

Model	Inputs	Output
pix2pix Generator	[32, 256, 256, 1]	[32, 256, 256, 1]
MI-pix2pix Generator	[32, 256, 256, 3]	[32, 256, 256, 1]
MI-GAN Generator	[32, 256, 256, 3]	[32, 256, 256, 1]

Table 4.2. Generators input and output shapes used during training. 32 is the batch size we choose to use. The last input dimension is the number of channels, so it represents how many modalities can be fed to the model.

Model	Inputs	Output
pix2pix Discriminator	[32, 256, 256, 1],[32, 256, 256, 1]	[32, 30, 30, 1]
MI-pix2pix Discriminator	[32, 256, 256, 3],[32, 256, 256, 1]	[32, 30, 30, 1]
MI-GAN Discriminator	[32, 256, 256, 3],[32, 256, 256, 1]	[32, 15, 15, 1]

Table 4.3. Discriminators input and output shapes used during training.

Python 3.6.9 is the main programming language of our work. We used [33] as reference for the implementation of the pix2pix model that, as MI-pix2pix and MI-GAN, was built using mainly the library of Tensorflow 2.1.0. For our experiments we used Google Colaboratory, a hosted Jupyter notebook service, that offers free usage of its computing hardware consisting of a Tesla P100-PCIE-16GB GPU with 26 GB of RAM available and an Intel(R) Xeon(R) CPU @ 2.30GHz.

4.4 Evaluation Metrics

In subsection 4.3.2 we discussed about how much is important to evaluate an image in a qualitatively way, by looking not only at the losses values reached by the networks through the epochs in the training phase but also at the quality of the generated samples. Beside the qualitative measure though, it was necessary to define some quantitative metrics that were able to understand how much error and similarity there were between our prediction and the original image.

These metrics resulted particularly useful during training when there wasn't any visible difference in the quality of the synthesized modalities between two epochs as well as, on the other hand, human perception was very effective in the cases of images, from different epochs, with almost same values but with a clear difference in low-level details that the metrics couldn't capture.

4.4.1 Evaluation of the Whole Image

The first metrics we implemented were the ones to evaluate the whole area of the images produced by the GANs. In order to obtain results as more accurate as possible, we proceeded to crop and normalize both the prediction and the original image: we center-cropped the generated images of size 256x256 to the dimensions of 155x194, after calculating the largest bounding box able to contain each brain of the dataset [1].

This was motivated by the fact that computing, for example, the similarity between two images padded to have compatible size with the input of the networks returns a much higher value than the one obtained comparing two 155x194 slices: an high value due to the presence of many black pixels that risks to bias the performances on the metrics (Figure 4.10).

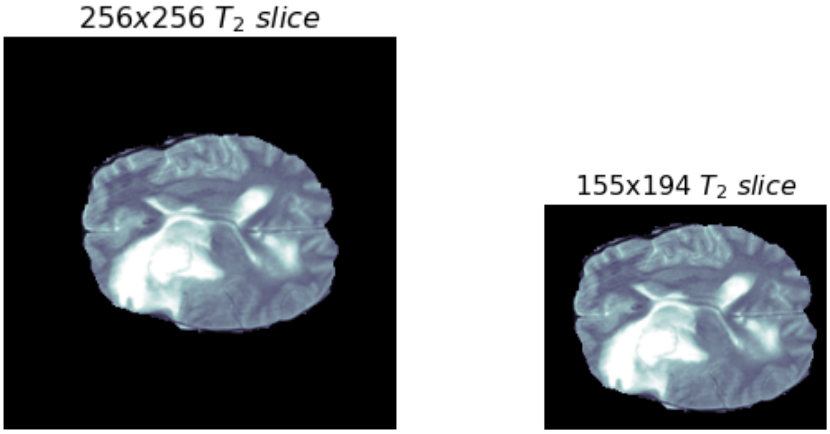


Figure 4.10. $T2$ slice before and after crop to 155x194, applied to remove as many not relevant pixels as possible before the calculation of the metrics.

In order to obtain consistent results during the comparison and avoid to compute a metric between an original image that has intensity values ranging from 0 to 1 and a predicted image with a different range, happening especially during the evaluation of the model in the first epochs of the training (while at the end of the training the GAN learns to synthesize images that lie in the same normalized range of the input image), we proceeded to apply a scaling.

The transformation applied to the original image and to its prediction is a **mean normalization**, whose formula, taken from [35], is the following:

$$x' = \frac{x - \text{average(x)}}{\max(x) - \min(x)} \tag{4.2}$$

where x is the original value, x' is the normalized value. This method, similar to the min-max, allows the pixels to have approximately zero mean and dynamic range equal to 1.

Differently from the normalization done, in the preprocessing step (Section 3.4), to the whole volume, here the transformation is applied with respect to the slice (the min and max values are respectively the minimum and maximum intensity values of the slice, not of the whole volume). We didn't normalize with respect to the volume because our GANs are trained to predict a single image based on a single slice: therefore we considered this approach as a better method to evaluate our predictions.

Mean Squared Error

Mean Squared Error (MSE), one of the metrics that we implemented, allows to estimate the error between its inputs. More specifically, MSE measures the average squared difference between the estimated pixels values and the intensity values belonging to the true image. It is computed as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_i')^2 \tag{4.3}$$

where y_i is the original image while y_i' is the synthesized image.

To implement it, we used a function from the library of Tensorflow that computes a MSE per each pixel in the image (that is the same of doing just a squared difference between the true pixel and the predicted pixel, since N = 1). The function returns a matrix 155x194, whose values are then averaged to obtain the final mean squared error between y_i and y_i'.

Peak Signal-to-Noise Ratio

Peak signal-to-noise ratio, often abbreviated PSNR, is a metric that computes, in decibels (dB), the ratio between the maximum intensity value allowed in the image [36] and the MSE and is defined as:

$$PSNR = 10 \log_{10} \frac{I_{max}^2}{MSE} \tag{4.4}$$

where I_{max} is the dynamic range that corresponds to the maximum intensity value of the image when its range is [0, 1]. The higher the PSNR, the better is the quality of the predicted image, since the ratio has a measure of error, the MSE, as denominator. For the implementation of this metric we used a built-in function from Tensorflow that receives two images and directly computes the PSNR value.

Structural Similarity

This metric is defined as the structural similarity (SSIM) index between two images, designed to improve on traditional metrics such as Mean Squared Error and Peak Signal-to-Noise Ratio. SSIM ranges from 0 to 1 and is a metric that doesn't compare the images just on a single statistic: its value of similarity is based on different factors such as the luminance, the contrast and the structure. Furthermore, the structural similarity is applied locally over the image due to the fact that pixels spatially close are the ones with stronger inter-dependencies, somehow similar to the way in which human perception works. SSIM is computed as:

$$SSIM = \frac{(2\mu_x + \mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{4.5}$$

where x, y are the two images to be compared, μ is the mean intensity, σ^2 the variance of the image and σ_{xy} the covariance of x, y.

For calculating this metric we used the implementation provided by the Tensorflow library which in turn is based on the standard SSIM implementation from [37]. The only parameter that we needed to specify was the dynamic range of the images (difference between the max and min allowed values), that in our case, thanks to the mean normalization, is equal to 1.

The SSIM reported in Chapter 5 are obtained by computing the mean and standard deviation over all the SSIM values calculated by comparing each original image in the test test with its prediction.

4.4.2 Evaluation of the Tumor Area

The most relevant part of the brain, to us and in general to the doctors that have to make a diagnosis, is the tumor. Because of this we implemented three more ways in which we could evaluate the results of our predictions, not anymore by looking at the whole brain area but by focusing on the tumor.

The exact information of which pixels belong to the tumor and which do not is contained in the *Ground Truth* volumes that show the segmentation of the malignant area of the brain.

MSE and PSNR

To compute the performances on the tumor area we implemented a modified version of the metrics used to evaluate the whole area: MSE and PSNR. To do this, we had first to use the ground truth as a mask to isolate the interested area by setting to 0 all the pixels belonging to the healthy part of the brain. Figure 4.11 illustrates the result obtained by masking a T_{2flair} slice with its ground truth.

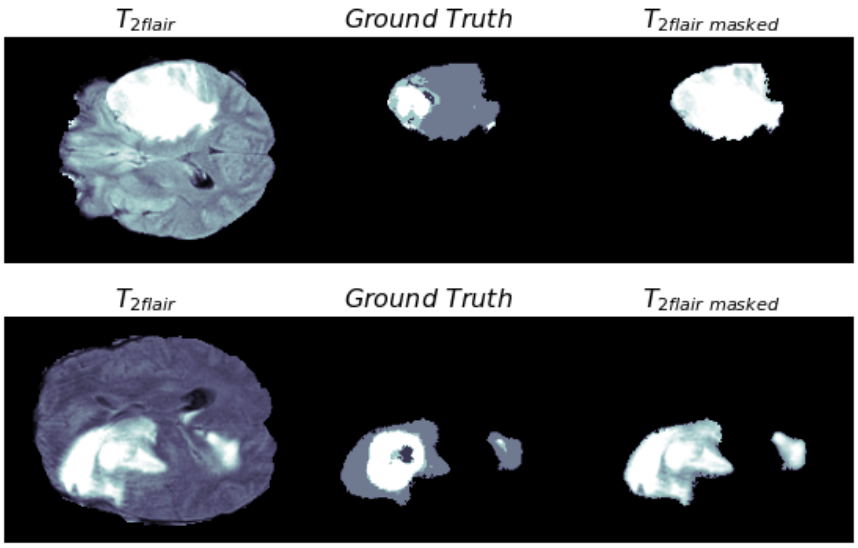


Figure 4.11. T_{2flair} slices and their masked versions, using the ground truth.

After scaling with mean normalization (4.2) and masking both the original image and the predicted missing modality with the ground truth, we proceeded to measure the quality of the images with the modified versions of MSE and PSNR.

The idea was to modify the MSE (and so the PSNR, that depends on the MSE) by including in the calculation only the pixels belonging to the tumor area, so without considering the pixels corresponding to the back pixels in the ground truth image.

It was necessary to not consider all these black pixels because they would have

affected (and boosted) the metrics of similarity, by reaching higher values, as well as the error metrics, by dropping significantly to values close to 0.

We then computed the MSE between the masked images (target masked and prediction masked) by dividing the squared difference by the number of pixels in the ground truth with value different from 0, instead of dividing by the total number of pixels in the whole image. It was sufficient to apply only this slightly variation to the MSE Formula (4.3), since the squared difference value wasn't affected by the presence of the black pixels: both the masked images had zero intensity in the dark area, resulting in a difference of values equal to 0.

It is also important to highlight that, even if every patient has a tumor, not all the slices extracted from a volume contain a malignant area. Because of this, we didn't include in the calculations the slices whose masked images were completely black, since we are interested in evaluating only the generated area on the tumor: keeping all the masked slices would have boosted, again, the metrics.

Dice Similarity Coefficient

To understand the quality of the generation process, beside the human perception and the other quantitative metrics implemented, we proceeded to evaluate our generated images using segmentation models that extract the tumor area from the synthesized missing modalities. In particular, we tested the result of the T_{2flair} segmentation model from [38] using as input the T_{2flair} predictions (with dimensions 256x256) obtained from our trained models (pix2pix, MI-pix2pix and MI-GAN).

The metric we implemented to evaluate the tumor segmentation model is the Dice similarity coefficient (DSC), also known as Sørensen–Dice coefficient, that measures the similarity between two images. DSC ranges from 0 to 1 and is defined as:

$$DSC = \frac{2TP}{2TP + FP + FN} \tag{4.6}$$

where TP (True positives) represents the number of pixels in the segmentation image correctly classified as belonging to the tumor while FP (False positives) is the number of pixels misclassified as positive. FN (False negatives) is instead the number of positive pixels in the ground truth that are classified as negative in the segmentation. To implement this metric, we first counted the TP, FP, FN per every subject and from these values we calculated the DSC with respect to the volume, using 4.6, and then we averaged the computed values of the 28 volumes (from the test set) to obtain the final score, reported in Chapter 5. Figure 4.12 shows the segmentation

obtained from T_{2flair} slices using a segmentation model [38]: the DSC is computed between the ground truth and the segmentation.

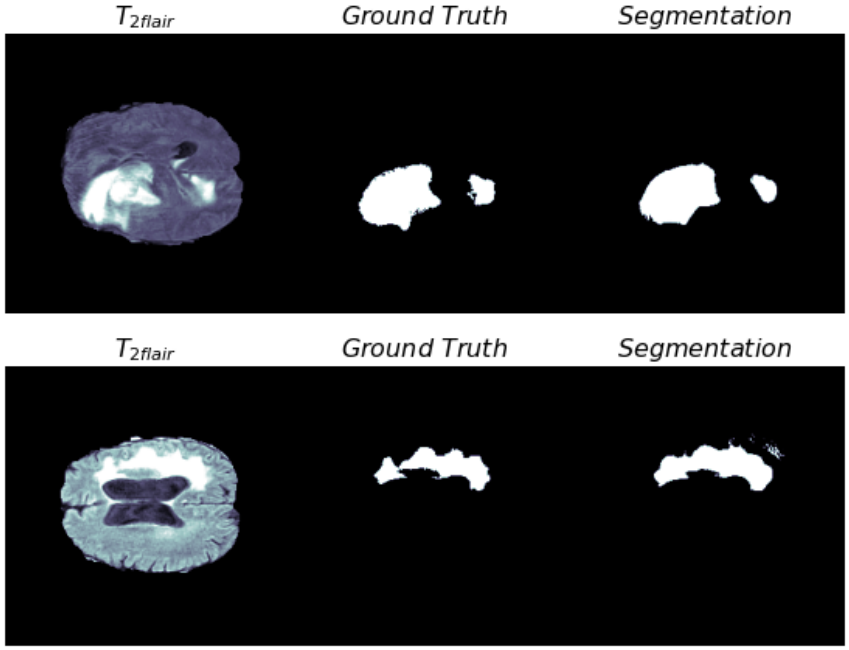


Figure 4.12. Comparison between Ground Truths and segmentations of T_{2flair} slices obtained using the segmentation model from [38].

4.5 Summary

In this chapter we first discussed about the input pipeline optimization of our models. We proceeded then to present a detailed description about the architectures of the models we implemented: pix2pix, MI-pix2pix, MI-GAN.

We gave an overview of the training algorithm, of the losses used in the update step and the implementation details of this work in order to allow the reproducibility of our results.

At the end of the chapter is described how we evaluated the synthesized missing modalities obtained from the GANs in order to measure, as accurately as possible, the quality of the generation in both the tumor area and the whole area of the brain.

Chapter 5

Results

In this chapter we report the results from our work: in Section 5.1 are summarized the quantitative performances obtained by all the model trained, while in Section 5.2 we illustrate, in a qualitative way, the ability of our GANs to synthesize MRI sequences, using the modalities contained in BRATS2015.

In Section 5.3 we provide our discussion on the results by considering them in the same order they are presented.

5.1 Quantitative Results

In this section we show the quantitative results reached by all the models we experimented with. In total we trained 14 models using the three GANs presented in Chapter 4. Subsection 5.1.1 shows the results obtained by four of these models, trained to generate T_1 slices. In Subsection 5.1.2 we present the ones relative to the generation of T_2 while Subsections 5.1.3 and 5.1.4 contain the performances of the generative models trained to synthesize, respectively, T_{1c} and T_{2flair} slices. The tables reported in these subsections present three metrics used for the evaluation of the whole image, MSE, PSNR and SSIM, but also $\text{MSE}_{\text{tumor}}$ and $\text{PSNR}_{\text{tumor}}$, computed considering only the generated tumor area.

The pix2pix models (Subsection 4.2.1) are called P2P (where P2P($T_{1c\to1}$) is trained to generate T_1 from T_{1c}) while, for example, MI-P2P$_{T_2}$ is the MI-pix2pix GAN (Subsection 4.2.2) generating T_2 from (T_1, T_{1c}, T_{2flair}).

Moreover, we introduced some baselines in order to understand if our single-input models were actually learning to generate a missing modality y' or if, instead, they were only exploiting the similarities between the input and the output: basically we wanted to evaluate if there was more similarity between the generated image y' and y

than between x and y, in order to understand if the GAN was correctly elaborating the image received in input. Therefore, Baseline(x, y) corresponds to the values calculated by applying the metrics between modality x and modality y.

In the tables of the following subsections, values in boldface represent best performance values. The reported values are mean ± std.

5.1.1 Generation of T_1

Four models were trained to generate the missing modality T_1: two unimodal GANs with different inputs (T_2 and T_{1c}) and two multimodal models. We choose to train different pix2pix models to understand how much good could have performed a single-input network that receives an image highly similar to the target (T_{1c}) compared to P2P ($T_{2\to1}$). The results are presented in Table 5.1.

Model	**MSE**	**PSNR**	**SSIM**	**MSE$_{\text{tumor}}$**	**PSNR$_{\text{tumor}}$**
Baseline ($T_{2,1}$)	0.0396 ± 0.0275	15.3286 ± 4.2134	0.5054 ± 0.2116	0.0594 ± 0.0523	13.6678 ± 3.6085
P2P ($T_{2\to1}$)	0.0060 ± 0.0046	23.4967 ± 3.6754	0.8112 ± 0.1004	0.0199 ± 0.0187	18.5047 ± 3.7607
Baseline ($T_{1c,1}$)	0.0058 ± 0.0050	23.8431 ± 4.0912	0.8096 ± 0.0984	0.0173 ± 0.0216	20.1544 ± 5.0543
P2P ($T_{1c\to1}$)	0.0044 ± 0.0041	25.0680 ± 3.8652	0.8403 ± 0.0856	0.0114 ± 0.0143	21.4485 ± 4.3380
MI−P2P$_{T_1}$	0.0044 ± 0.0040	24.9339 ± 3.6983	0.8413 ± 0.0838	0.0113 ± 0.0099	20.8938 ± 3.6111
MI−GAN$_{T_1}$	$\mathbf{0.0041 \pm 0.0038}$	$\mathbf{25.2569 \pm 3.6512}$	$\mathbf{0.8472 \pm 0.0830}$	$\mathbf{0.0102 \pm 0.0097}$	$\mathbf{21.5359 \pm 3.8620}$

Table 5.1. Generation of T_1: performances on the test set.

5.1.2 Generation of T_2

Also for the generation of T_2 we opted for training four models where two of these are unimodal GANs: the first one is a pix2pix trained to receive T_1 as input while the second one takes as input the T_{2flair} modality, that captures more similar characteristics (of T_2) than T_1. The results are presented in Table 5.2.

Model	**MSE**	**PSNR**	**SSIM**	**MSE$_{\text{tumor}}$**	**PSNR$_{\text{tumor}}$**
Baseline ($T_{1,2}$)	0.0396 ± 0.0275	15.3286 ± 4.2134	0.5054 ± 0.2116	0.0594 ± 0.0523	13.6678 ± 3.6085
P2P ($T_{1\to2}$)	0.0100 ± 0.0074	21.3182 ± 3.8023	0.7521 ± 0.1247	0.0476 ± 0.0397	14.3652 ± 3.3523
Baseline ($T_{2f,2}$)	0.0275 ± 0.0199	16.8268 ± 3.9727	0.6262 ± 0.1597	0.0464 ± 0.0500	15.1591 ± 4.0428
P2P ($T_{2flair\to2}$)	0.0087 ± 0.0076	21.9227 ± 3.7021	0.7567 ± 0.1287	0.0256 ± 0.0242	17.3035 ± 3.4584
MI−P2P$_{T_2}$	$\mathbf{0.0073 \pm 0.0063}$	$\mathbf{22.7645 \pm 3.8272}$	$\mathbf{0.8005 \pm 0.1112}$	0.0207 ± 0.0167	$\mathbf{18.1305 \pm 3.5930}$
MI−GAN$_{T_2}$	0.0077 ± 0.0061	22.3719 ± 3.5290	0.7835 ± 0.1141	$\mathbf{0.0205 \pm 0.0167}$	18.0725 ± 3.3763

Table 5.2. Generation of T_2: performances on the test set.

5.1.3 Generation of T_{1c}

Table 5.3 shows instead the performances obtained by three models trained to generate T_{1c} slices: one pix2pix model and two multi-input models. We choose T_1 as input modality for the unimodal GAN because it is the more similar sequence, among the ones available, to the target T_{1c}.

Model	**MSE**	**PSNR**	**SSIM**	**MSE$_{\text{tumor}}$**	**PSNR$_{\text{tumor}}$**
Baseline ($T_{1,1c}$)	0.0058 ± 0.0050	23.8431 ± 4.0912	0.8096 ± 0.0984	0.0173 ± 0.0216	20.1544 ± 5.0543
P2P ($T_{1\to1c}$)	$\mathbf{0.0051 \pm 0.0048}$	$\mathbf{24.6165 \pm 4.0755}$	$\mathbf{0.8139 \pm 0.0996}$	$\mathbf{0.0155 \pm 0.0199}$	$\mathbf{20.6981 \pm 4.9762}$
MI−P2P$_{T_{1c}}$	0.0052 ± 0.0040	24.1597 ± 3.8631	0.8110 ± 0.0963	0.0168 ± 0.0172	19.8441 ± 4.6258
MI−GAN$_{T_{1c}}$	0.0054 ± 0.0040	23.9242 ± 3.6958	0.8027 ± 0.1003	0.0157 ± 0.0162	19.9779 ± 4.3568

Table 5.3. Generation of T_{1c}: performances on the test set.

5.1.4 Generation of T_{2flair}

Tables 5.4 and 5.5 summarize the performances reached by the models synthesizing T_{2flair}: in particular 5.5 shows the additional metric we implemented to evaluate the quality of the generated tumor area using a single-input segmentation model from [38]. The DSC is calculated between the ground truth and the segmentation of our synthesized images. Furthermore, a reference DSC score is computed between the ground truth and the segmentation of the original slice from BRATS2015.

As in Subsection 5.1.3, also here we choose to train only one pix2pix model, using as input T_2 that is the more similar sequence, among the ones available, to the target.

Model	MSE	PSNR	SSIM	$\text{MSE}_{\text{tumor}}$	$\text{PSNR}_{\text{tumor}}$
Baseline ($T_{2,2f}$)	0.0275 ± 0.0199	16.8268 ± 3.9727	0.6262 ± 0.1597	0.0464 ± 0.0500	15.1591 ± 4.0428
P2P ($T_{2\to 2flair}$)	0.0090 ± 0.0065	21.5895 ± 3.4831	0.7518 ± 0.1211	0.0390 ± 0.0463	15.9946 ± 4.0459
MI−P2P$_{T_{2flair}}$	$\mathbf{0.0069 \pm 0.0049}$	$\mathbf{22.8165 \pm 3.7317}$	$\mathbf{0.7772 \pm 0.1094}$	$\mathbf{0.0221 \pm 0.0375}$	$\mathbf{19.0374 \pm 4.1582}$
MI−GAN$_{T_{2flair}}$	0.0072 ± 0.0050	22.5524 ± 3.5655	0.7610 ± 0.1175	0.0258 ± 0.0285	17.4694 ± 3.6137

Table 5.4. Generation of T_{2flair}: performances on the test set.

Segmentation image	$\text{DSC}_{\text{tumor}}$
Original T_{2flair}	0.8053 ± 0.1156
P2P($T_{2\to 2flair}$)	0.6632 ± 0.1608
MI-P2P$_{T_{2flair}}$	$\mathbf{0.7427 \pm 0.1810}$
MI-GAN$_{T_{2flair}}$	0.6837 ± 0.2136

Table 5.5. DSC performances, on the test set, obtained by comparing the ground truth and the segmentations, using segmentation model in [38], of T_{2flair} predictions.

Additionally, in Table 5.6 we report the time, in epochs, taken to train our models.

Model	Training time (epochs)
P2P($T_{2\to 1}$)	43
P2P($T_{1c\to 1}$)	55
MI-P2P$_{T_1}$	40
MI-GAN$_{T_1}$	30
P2P($T_{1\to 2}$)	70
P2P($T_{2flair\to 2}$)	25
MI-P2P$_{T_2}$	35
MI-GAN$_{T_2}$	37
P2P($T_{1\to 1c}$)	42
MI-P2P$_{T_{1c}}$	50
MI-GAN$_{T_{1c}}$	40
P2P($T_{2\to 2flair}$)	27
MI-P2P$_{T_{2flair}}$	35
MI-GAN$_{T_{2flair}}$	25

Table 5.6. Training time (in epochs) of the 14 models trained in this work.

5.2 Qualitative Results

This sections presents the qualitative results from the missing modalities synthesized by the networks (5.2.1) and the segmentation results using our predictions (5.2.2).

5.2.1 Generated Samples

Figures 5.1 and 5.2 illustrate the differences between the pix2pix models we trained to generate, respectively, T_1 and T_2 slices.

In Figures 5.3, 5.4, 5.5, 5.6 are shown the generated images outputted by the models trained to synthesize, respectively, T_1, T_2, T_{1c} and T_{2flair}. Figures 5.3 and 5.4 contain predictions from MI-GAN, MI-pix2pix and only one of the two pix2pix models trained: we choose the one with more competitive results, compared to the other model.

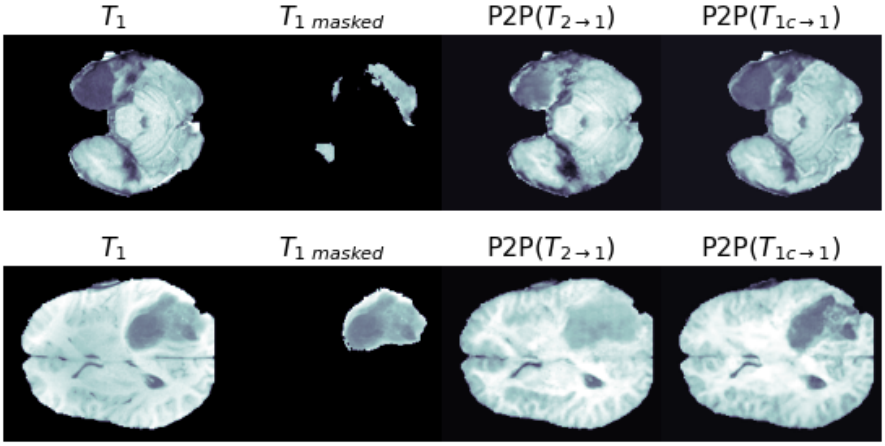


Figure 5.1. P2P($T_{2\to1}$) and P2P($T_{1c\to1}$) predictions in two different subjects from the test set. $T_{1masked}$ is the original slice masked with the ground truth.

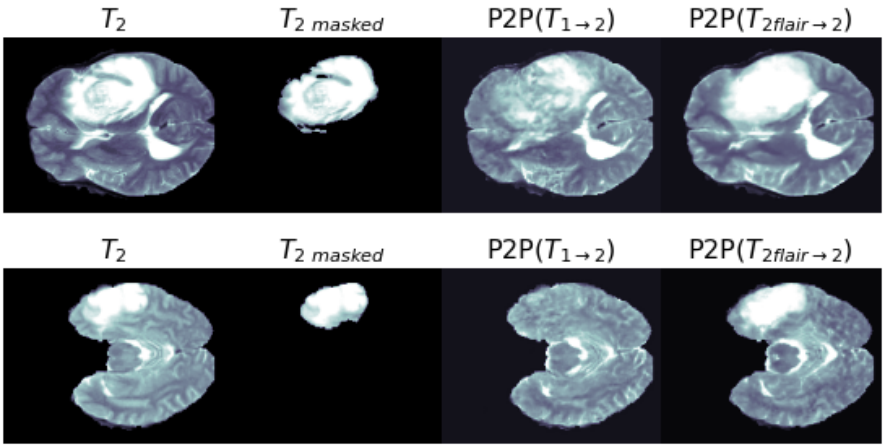


Figure 5.2. P2P($T_{1\to2}$) and P2P($T_{2flair\to2}$) predictions in two different subjects from the test. $T_{2masked}$ is the original slice masked with the ground truth.

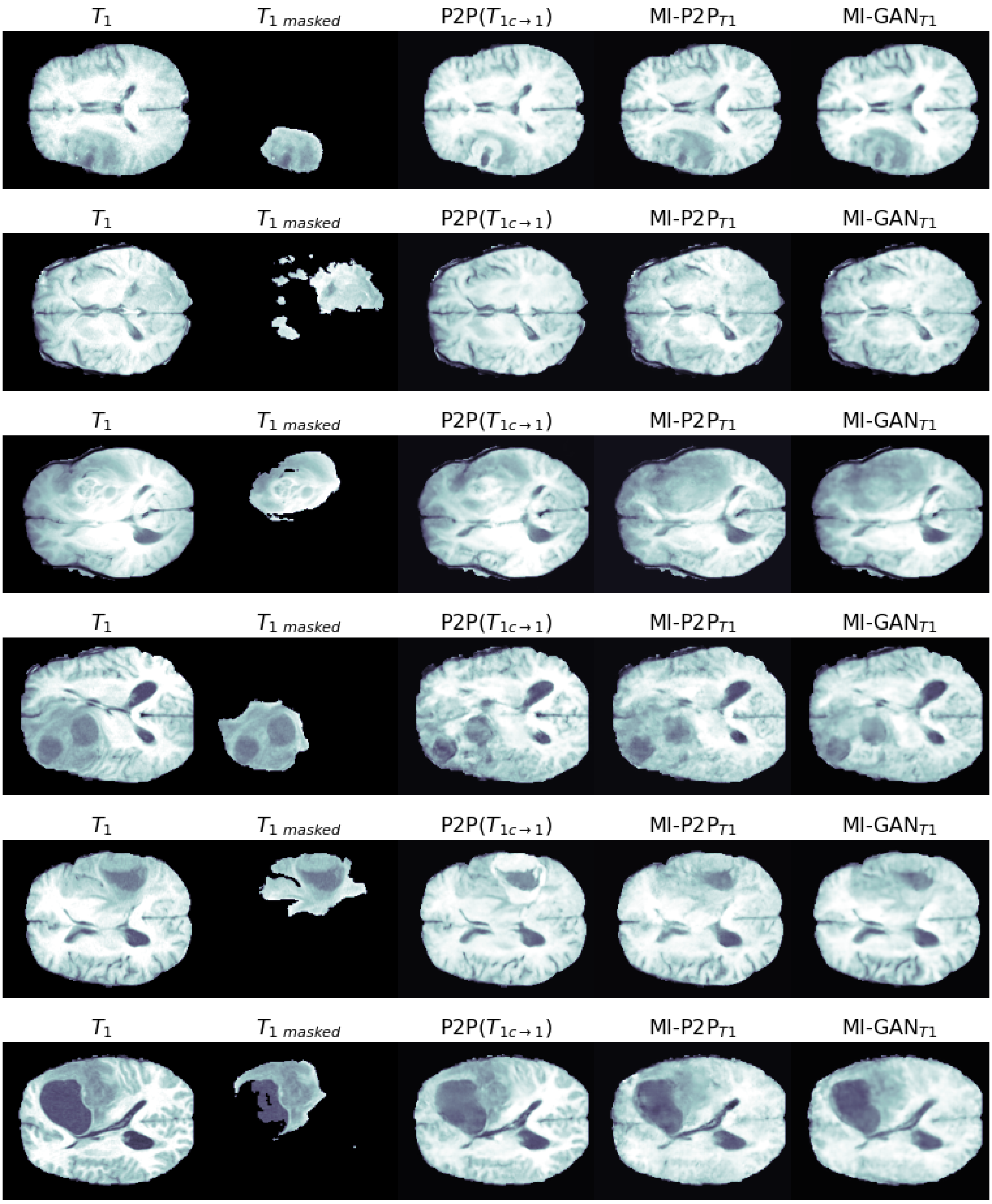


Figure 5.3. Comparison between the original slices from T_1 and its predictions using P2P($T_{1c \to 1}$), MI-P2P$_{T_1}$ and MI-GAN$_{T_1}$. Each row corresponds to a different subject from the test set.

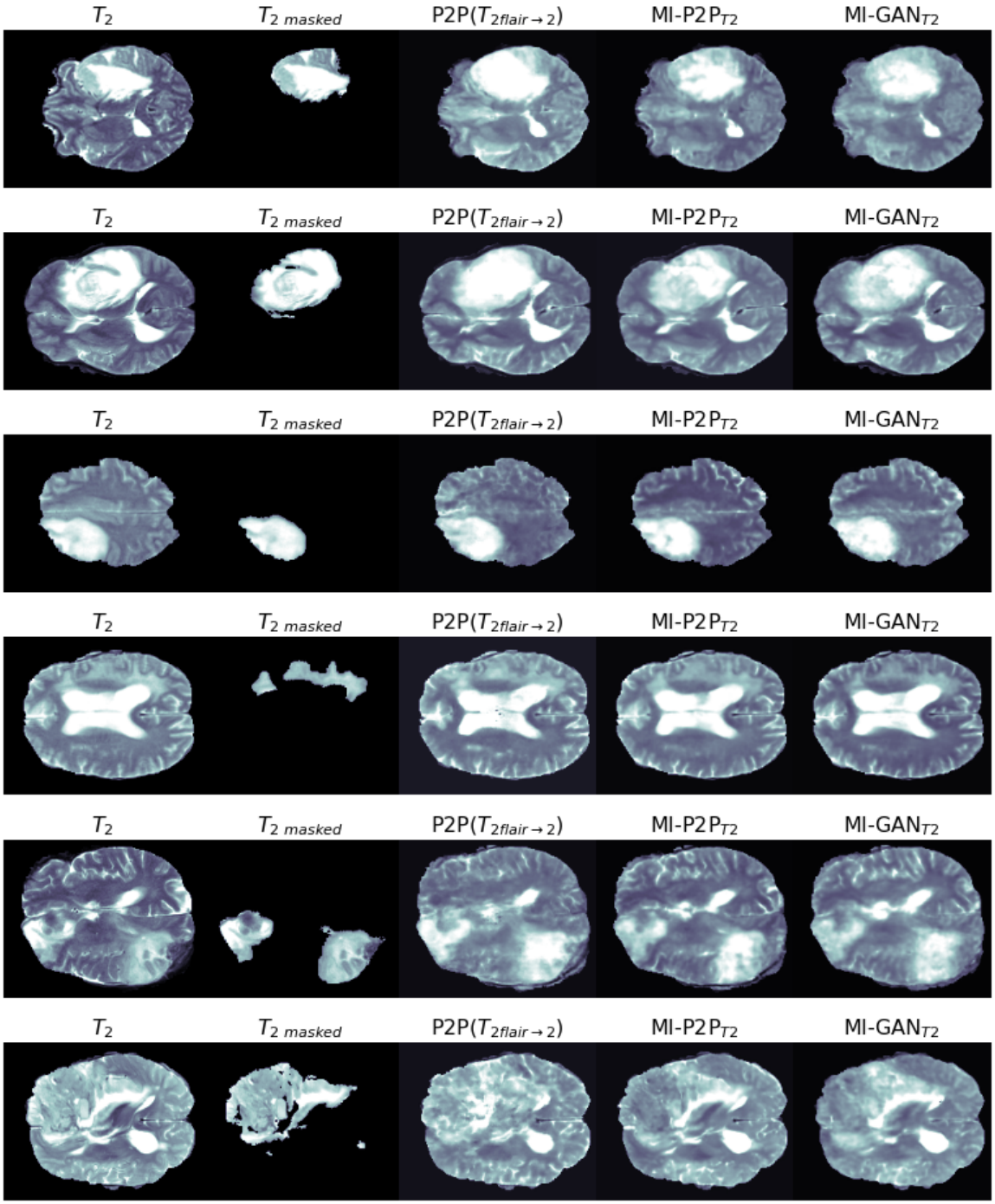


Figure 5.4. Comparison between the original slices from T_2 and its predictions using P2P($T_{2flair\rightarrow 2}$), MI-P2P$_{T_2}$ and MI-GAN$_{T_2}$. Each row corresponds to a different subject from the test set.

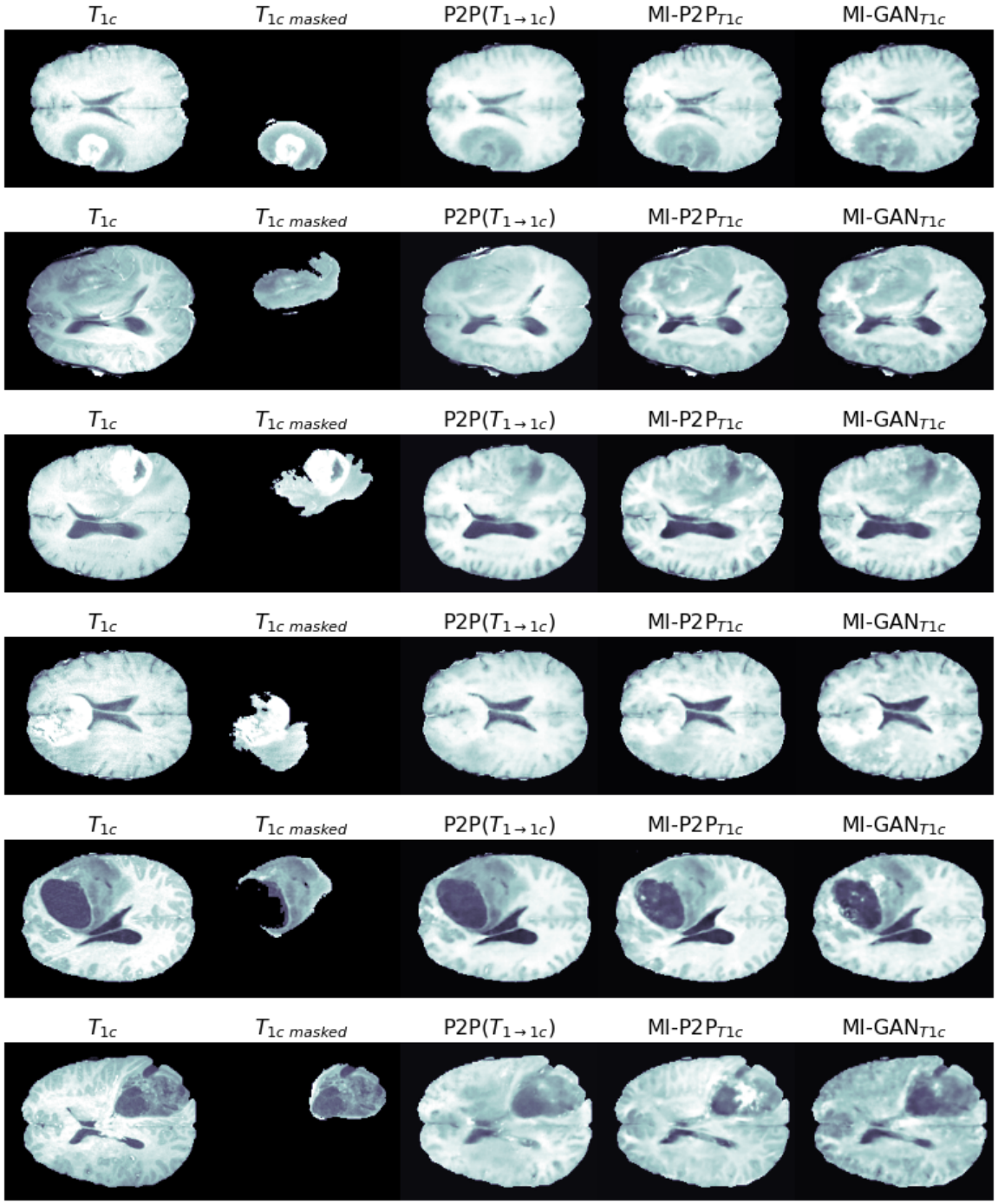


Figure 5.5. Comparison between the original slices from T_{1c} and its predictions, using P2P($T_{1\rightarrow 1c}$), MI-P2P$_{T_{1c}}$ and MI-GAN$_{T_{1c}}$. Each row corresponds to a different subject from the test set.

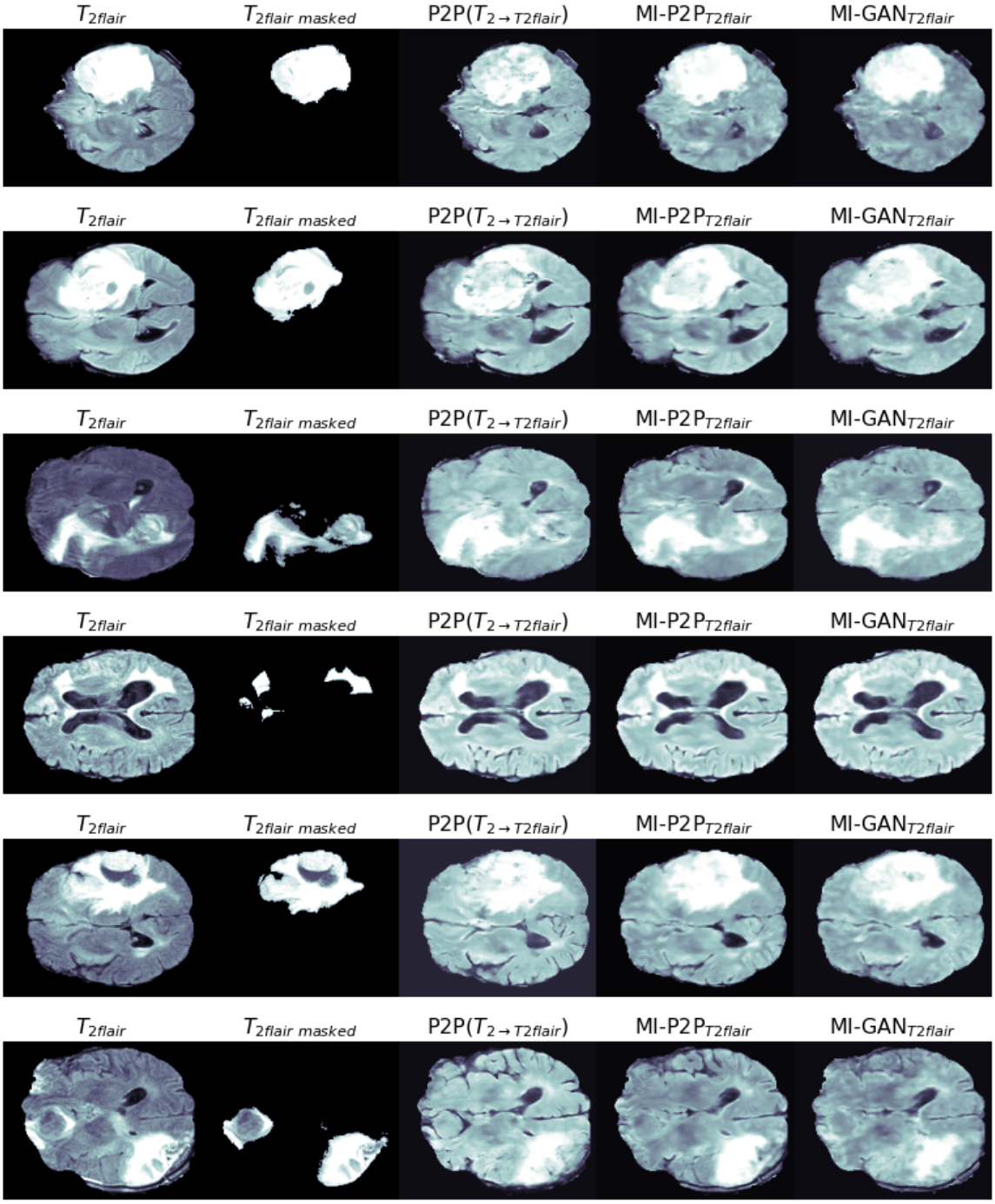


Figure 5.6. Comparison between the original slices from T_{2flair} and its predictions, using P2P($T_{2\rightarrow 2flair}$), MI-P2P$_{T_{2flair}}$ and MI-GAN$_{T_{2flair}}$. Each row corresponds to a different subject from the test set.

5.2.2 Segmentation using GAN Predictions

In Figure 5.7 are illustrated the results obtained by segmenting our synthesized missing modalities.

In particular we tested the T_{2flair} single-input segmentation model from [38] using the predictions outputted by P2P($T_{2\rightarrow 2flair}$), MI-P2P$_{T_{2flair}}$ and MI-GAN$_{T_{2flair}}$. We also show the segmentation obtained using the original T_{2flair} slice from BRATS2015 and the Ground Truth, representing the true tumor area segmented by a domain expert.

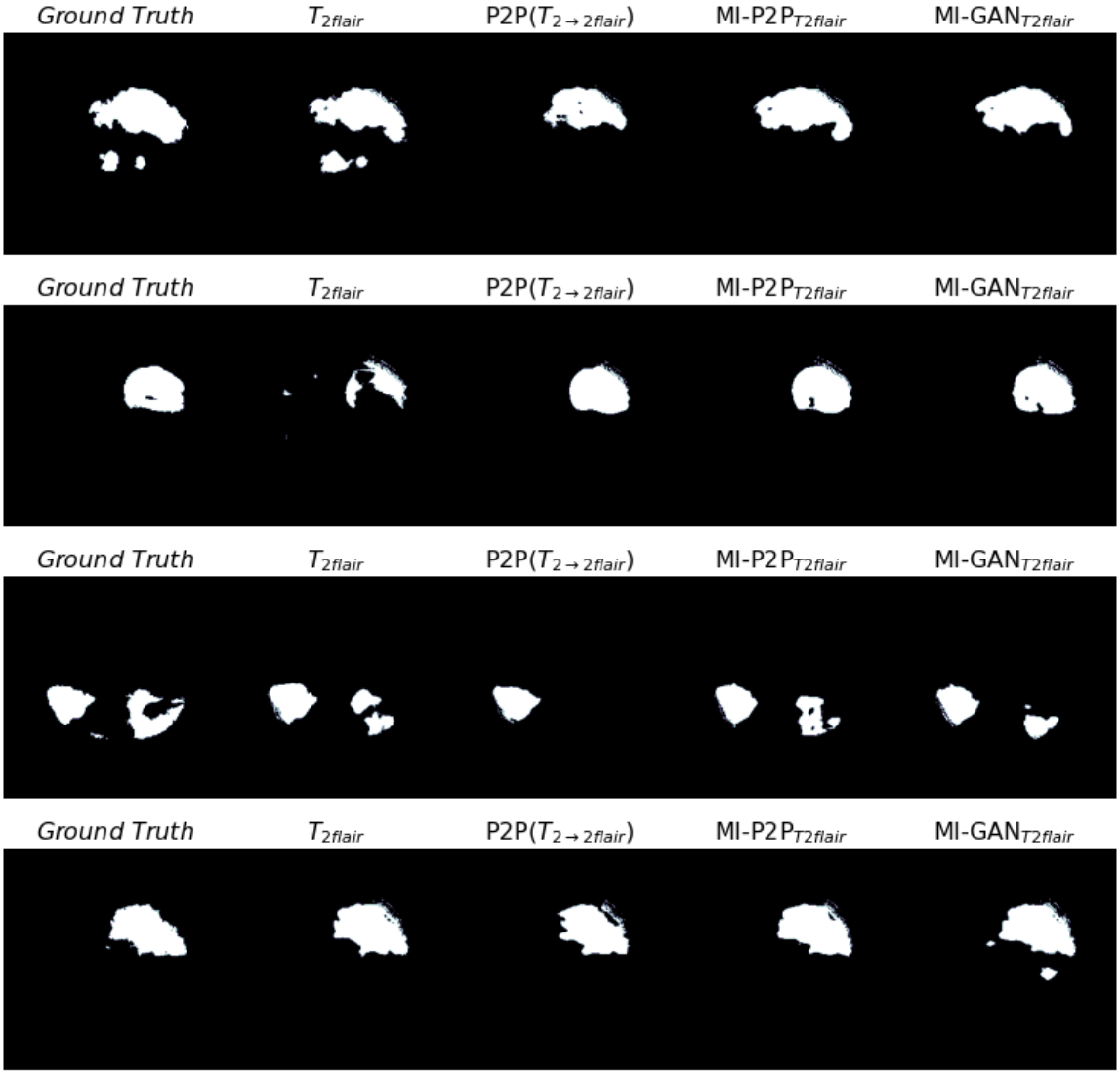


Figure 5.7. From left to right: ground truth and the segmentations from T_{2flair}, pix2pix, MI-pix2pix and MI-GAN. Each row corresponds to a different subject from the test set.

5.3 Results Evaluation

We conduct the analysis of the results in the same order they are presented in the previous sections. The first part of our discussion considers the quantitative results presented in Section 5.1. After that, we do some considerations on the qualitative results from the trained models, presented in Section 5.2.

5.3.1 Quantitative Results Discussion

Pix2pix vs Baseline

Tables 5.1, 5.2, 5.3, 5.4 confirm what we expected about the processing ability of the Generative Adversarial Networks implemented in this work: all the Baseline(x, y) scores reported are outperformed by the respective pix2pix models, that receive as input a modality x and generate the missing modality y'. For example P2P($T_{2\rightarrow 2flair}$) performs better by 35% in PSNR, 38% in SSIM and 560% in MSE with respect to Baseline(T_2, T_{2flair}), proving that our GANs aren't only outputting the input but they are extracting and processing the information of the image received, resulting in a higher similarity between the generated sample (y') and the ground truth (y) than between y and input x.

Single-Input vs Multi-Input Synthesis

We noticed that the multi-input models outperform the single-input models when the target missing modality are T_2 and T_{2flair} while the unimodal generative models of the other two sequences we experimented with reach very competitive results with respect to the two multi-modal GANs.

In particular, P2P($T_{1\rightarrow 1c}$) performs better than MI-GAN and MI-pix2pix in all the evaluation metrics considered: compared to the first model it shows improvement of 3% in PSNR, 1% in SSIM and 6% in MSE. These percentage increases can be attributed to the fact that T_{1c} and T_1 are very similar, as confirmed by the values obtained by calculating the evaluation metrics between the two modalities (Baseline(T_1, T_{1c})). It means that in order to generate T_{1c} slices, T_1 is the sequence with highest significance and the information that carries through the network is enough to produce a realistic synthesized image, without the contribution of T_2 and T_{2flair}. This was confirmed one more time by looking at Table 5.1 where P2P($T_{1c\rightarrow 1}$) has better performances compared to MI-P2P$_{T_1}$ and competitive results with respect to MI-P2P$_{T_1}$, in both the overall area of the brain and the tumor area.

Results from Tables 5.2 and 5.4 reinforce the hypothesis that multi-modal models are objectively better than uni-modal models except in the case in which a modality capturing highly similar characteristics to the ones of the target image is received as input by a single-input GAN.

MI-pix2pix outperformed pix2pix in the synthesis of both T_2 and T_{2flair} (also the other many-to-one model, MI-GAN, showed better results with respect to pix2pix, in all the evaluation metrics): in particular, in the generation of T_2, MI-P2P$_{T_{2flair}}$ outperformed P2P($T_{2flair \to 2}$) by 4% in PSNR, 5% in SSIM and 19% in MSE despite the fact that T_{2flair} is the closest sequence to T_2: this is because the input modality used, as confirmed by Baseline(T_2, T_{2flair}), doesn't contain enough information to produce, alone, a realistic image.

In Tables 5.1 and 5.2 we reported the performances scored by different pix2pix models. From the analysis we can confirm that, when the input modality is not as much informative as others, then the pix2pix model performs poorly, compared to all the other models. A clear example of this is represented in the synthesis of T_1, that is more similar to T_{1c} than to T_2: P2P($T_{2 \to 1}$) is outperformed by P2P($T_{1c \to 1}$) (by 7% in PSNR, 4% in SSIM, 27% in MSE, 16% in PSNR$_{tumor}$ and 43% in MSE$_{tumor}$) and by the multi-modal models.

MI-pix2pix vs MI-GAN

We can also notice that MI-pix2pix seems to perform slightly better than MI-GAN in all the modalities with the exception of T_1 where MI-GAN is the most competitive model: proof that the real discriminant in the performances is not the architecture (and the loss) used but whether the model is single-input (and which modality takes as input) or multi-input.

Whole image vs Tumor area

Furthermore, we observe that the results from the metrics computed on the tumor area are consistent with the values obtained by the evaluation of the whole image, even if they are not directly comparable, since, for example, PSNR$_{tumor}$ was calculated considering only the tumor area while PSNR takes into account both the brain area and the black pixels: so an higher value from PSNR doesn't mean that the overall quality of the generated brain is better than the synthesized tumor area, since the obtained score is boosted by the presence of black pixels in both the images used for the comparison.

DSC performances

From the DSC values in Table 5.5 we notice that the segmentations obtained by using predictions from MI-pix2pix reach good performances with respect to the original segmented slice (-8% in DSC_{tumor}): DSC ranges from a minimum value of 0 and a maximum value of 1 but it has to evaluated by comparing it to the result reached by the segmentation of original image. In this sense we are overall satisfied with the results.

Further comments

For what concerns the time spent to train the models, reported in 5.6, we observe that a more similar input image to the target doesn't always imply a faster convergence from the GAN and, as a consequence, a shorter training: indeed P2P($T_{1c\to1}$) took more time (55) than P2P($T_{2\to1}$) (43).

5.3.2 Visual Acknowledgment

The results discussed above indicate that unimodal GANs reach more competitive results when the input image used is the modality that presents the more similar characteristics to the ones of the target. This fact can be appreciated by a visual comparison of the predictions from the two models in Figures 5.1 and 5.2 where P2P($T_{1c\to1}$) and P2P($T_{2flair\to2}$) are able to synthesize a better tumor area than, respectively, P2P($T_{2\to1}$) and P2P($T_{1\to2}$).

Figures 5.3, 5.4, 5.5, 5.6 confirm the results observed through the evaluation metrics, even though we notice that in the case of the T_1 generation, where there isn't a model outperforming the others, it is objectively difficult to choose, qualitatively, one model over the others. For example, in 5.3, the multi-input predictions from the first subject (1st row) seem to be qualitatively better than the one outputted by single-input model. On the other hand, predictions from the last patient (6th row) show that pix2pix was able to generate an image richer of low-level details, less blurred and more similar to the original slice, compared to the multi-modal predictions.

Additionally, from rows 1 and 3 of Figure 5.5 we can notice that no one of the predictions were able to synthesize correctly the white area belonging to the tumor: this is due to the fact that the missing T_{1c} is a unique modality, obtained with the injection of a contrast agent into the body, and it has highly specific information that it is not present in any other sequences.

Our models demonstrated to be able to generalize well through the whole test set and the predictions generated show the capacity of these GANs to synthesize realistic missing modalities with good quality in the low-level details, even if with the presence of few blurred artifacts.

Figures 5.7 indicates that the results analysed in Table 5.5 are consistent with the segmentations produced using our predictions. As expected, the segmentation using the prediction from the most competitive model in the generation of T_{2flair} is not as good as the one produced using the original image but we observed some cases (5.7, 2nd row) in which the ground truth was more similar to the segmentation coming from our predictions than to T_{2flair} segmented: this is motivated by the fact that our predictions processed and incorporated information belonging to other modalities and not present in the original T_{2flair}.

5.4 Summary

In this chapter we presented our results reporting all the scores obtained by evaluating our models on the test set: we showed the performances of the model through all the evaluation metrics implemented.

Then, for providing a visual comparison of the generated images, we showed a set of predictions outputted by our GANs and, in addition, the outputs of the segmentation model from [38] that took as input our T_{2flair} predictions.

At the end of the chapter we discussed in detail the results, in both a quantitative and qualitative way and we made some considerations about the advantage of using multimodal networks instead of the unimodal ones in order to leverage the information contained in all the available sequences. Single-input though that demonstrated to reach competitive results, compared to the multi-input scenario, in the case in which is received as input an image that is highly similar with respect to the target sequence. By observing the results, we understood that the real discriminant in the performances was the number of sequences given as input to the models, rather than the architecture and loss used.

In the next chapter, we analyse our trained models by performing some experiments with the generator networks.

Chapter 6

Experiments

This chapter describes the experiments we conducted in order to understand how the information is processed by the GANs implemented: in particular the focus was to investigate how the information passes through the generator network during the generative process. Section 6.1 contains a detailed study about the skip connections while in Section 6.2 we describe the analysis performed to observe the internal connections behaviour in the generator. In Section 6.3 we discuss the results.

We performed the experiments using the models trained to generate T_{2flair}. The results shown in this chapter are relative to the analyses of the MI-pix2pix generator network (since MI-p2p was the most competitive model in the synthesis of T_{2flair}). Further results from the experiments with pix2pix and MI-GAN can be found in Appendix A.

6.1 Skip Connections Analysis

Two experiments were performed in this section: in the first one (Subsection 6.1.1) we observed what happens to the output of the generator by sequentially turning off the skip connections (setting all the values in the skips to 0), from the innermost to the outermost and viceversa, while in the second experiment (Subsection 6.1.2) we replaced, again sequentially, the information passing through the skips of our analysed generator, with input image A, with the content from the skips of another generator, with input B (where A and B, from the same modality, belong to different subjects).

Quantitative and qualitative results of the first experiment are presented in Table 6.1 and Figure 6.2 (where [1,1,1,1,0,0,0] means that the three outermost skip are off). Results from the second one are reported in Table 6.2 and Figure 6.3 where the configuration [AAAABBB] indicates that the generator was perturbed with image B

in the three outermost skip connections.

Figure 6.1 illustrates the setting in which the three outermost skip connections are turned off/perturbed (these connections are shown in red color), corresponding to the configurations mentioned above ([1,1,1,1,0,0,0]/[AAAABBB]).

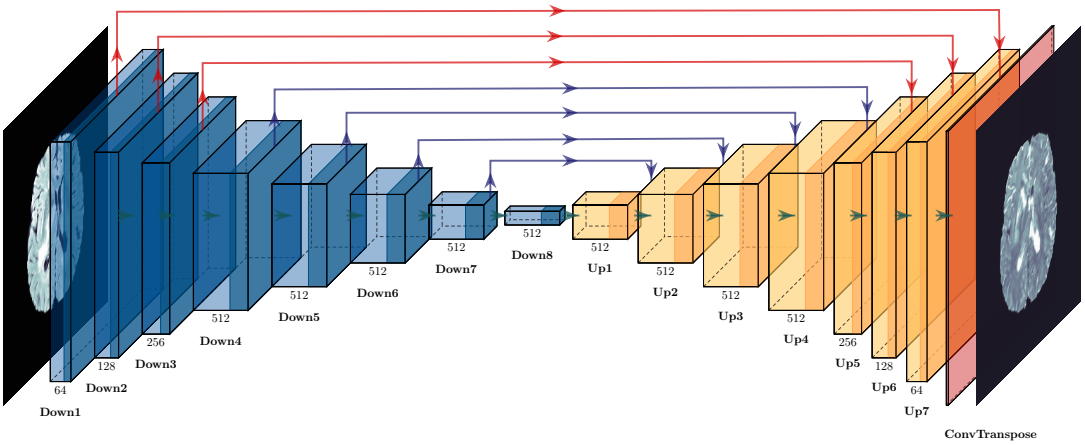


Figure 6.1. Example of network with the three outermost skip connections turned off/perturbed, corresponding to the configurations [1,1,1,1,0,0,0]/[AAAABBB].

6.1.1 Channels Turned Off

Skips off	**MSE**	**PSNR**	**SSIM**	**MSE$_{tumor}$**	**PSNR$_{tumor}$**
[1,1,1,1,1,1,1]	**0.0069 ± 0.0049**	**22.8165 ± 3.7317**	**0.7772 ± 0.1094**	0.0221 ± 0.0375	19.0374 ± 4.1582
[0,1,1,1,1,1,1]	0.0071 ± 0.0048	22.6609 ± 3.7374	0.7718 ± 0.1126	**0.0209 ± 0.0356**	**19.1225 ± 4.0804**
[0,0,1,1,1,1,1]	0.0093 ± 0.0053	21.3752 ± 3.7214	0.7520 ± 0.1163	0.0381 ± 0.0251	14.8111 ± 2.3423
[0,0,0,1,1,1,1]	0.0103 ± 0.0061	20.8808 ± 3.6404	0.7217 ± 0.1309	0.0374 ± 0.0438	15.5332 ± 3.1803
[0,0,0,0,1,1,1]	0.0105 ± 0.0059	20.8179 ± 3.7360	0.7423 ± 0.1175	0.0419 ± 0.0235	14.3061 ± 2.2416
[0,0,0,0,0,1,1]	0.0147 ± 0.0080	19.2806 ± 3.4188	0.6610 ± 0.1090	0.0779 ± 0.0231	11.3254 ± 1.6623
[0,0,0,0,0,0,1]	0.0116 ± 0.0060	20.1027 ± 2.8014	0.5940 ± 0.1035	0.0501 ± 0.0212	13.3909 ± 1.9644
[0,0,0,0,0,0,0]	0.0987 ± 0.0319	10.3797 ± 1.8937	0.0202 ± 0.0267	0.3230 ± 0.1115	5.1401 ± 1.4091
[1,1,1,1,1,1,1]	**0.0069 ± 0.0049**	**22.8165 ± 3.7317**	**0.7772 ± 0.1094**	**0.0221 ± 0.0375**	**19.0374 ± 4.1582**
[1,1,1,1,1,1,0]	0.0105 ± 0.0070	20.8009 ± 3.3278	0.7434 ± 0.1150	0.0332 ± 0.0556	17.3071 ± 4.3356
[1,1,1,1,1,0,0]	0.0533 ± 0.0256	13.4927 ± 3.1335	0.3956 ± 0.1498	0.1087 ± 0.0973	10.8014 ± 3.0530
[1,1,1,1,0,0,0]	0.0846 ± 0.0320	11.1146 ± 1.9886	0.1106 ± 0.0700	0.2253 ± 0.1313	7.0131 ± 2.0886
[1,1,1,0,0,0,0]	0.0978 ± 0.0336	10.4303 ± 1.8611	0.0413 ± 0.0491	0.3125 ± 0.1390	5.3861 ± 1.6662
[1,1,0,0,0,0,0]	0.1050 ± 0.0333	10.0808 ± 1.7531	0.0134 ± 0.0411	0.3362 ± 0.1386	5.0176 ± 1.5175
[1,0,0,0,0,0,0]	0.0963 ± 0.0315	10.4584 ± 1.7412	0.0255 ± 0.0419	0.3061 ± 0.1227	5.4125 ± 1.4862
[0,0,0,0,0,0,0]	0.0987 ± 0.0319	10.3797 ± 1.8937	0.0202 ± 0.0267	0.3230 ± 0.1115	5.1401 ± 1.4091

Table 6.1. Quantitative results from the Skip Connections Analysis on MI-P2P$_{T_{2flair}}$: skips are sequentially turned off.

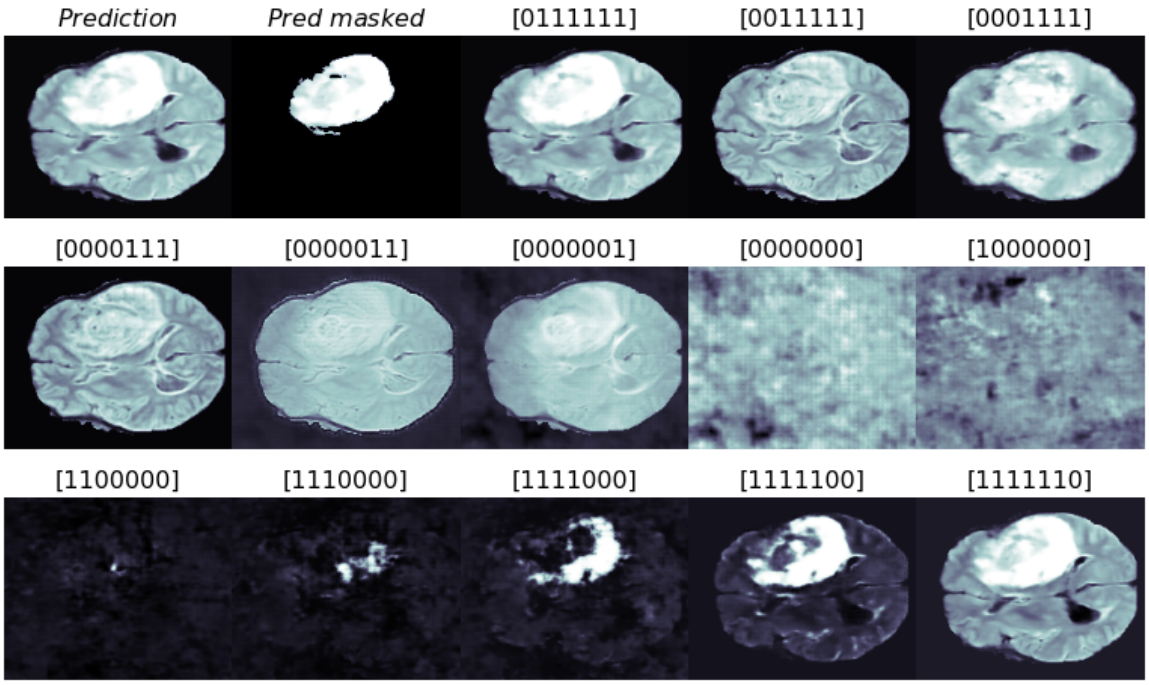


Figure 6.2. Qualitative results from the Skip Connections Analysis on MI-P2P$_{T_{2flair}}$: skips are sequentially turned off. *Prediction* corresponds to config.[1111111]

6.1.2 Channels Perturbed

Img.in Skips	MSE	PSNR	SSIM	MSE$_{\text{tumor}}$	PSNR$_{\text{tumor}}$
[AAAAAAA]	$\mathbf{0.0069 \pm 0.0049}$	$\mathbf{22.8165 \pm 3.7317}$	$\mathbf{0.7772 \pm 0.1094}$	0.0221 ± 0.0375	$\mathbf{19.0374 \pm 4.1582}$
[BAAAAAA]	0.0071 ± 0.0049	22.6889 ± 3.7357	0.7753 ± 0.1100	$\mathbf{0.0213 \pm 0.0333}$	18.9774 ± 4.0355
[BBAAAAA]	0.0090 ± 0.0054	21.5192 ± 3.6746	0.7539 ± 0.1145	0.0287 ± 0.0330	16.7935 ± 3.5473
[BBBAAAA]	0.0113 ± 0.0070	20.5749 ± 3.6649	0.7329 ± 0.1240	0.0429 ± 0.0359	14.8824 ± 3.4989
[BBBBAAA]	0.0146 ± 0.0085	19.2723 ± 3.2735	0.7045 ± 0.1187	0.0521 ± 0.0425	14.0021 ± 3.4752
[BBBBBAA]	0.0372 ± 0.0172	14.9083 ± 2.6436	0.5024 ± 0.1345	0.1032 ± 0.0885	11.6955 ± 4.6158
[BBBBBBA]	0.0599 ± 0.0282	12.9927 ± 3.1396	0.4099 ± 0.1852	0.1313 ± 0.0917	9.9851 ± 3.5551
[BBBBBBB]	0.0921 ± 0.0406	10.9644 ± 2.5820	0.3257 ± 0.1812	0.1871 ± 0.1677	9.0777 ± 4.4095
[AAAAAAA]	$\mathbf{0.0069 \pm 0.0049}$	$\mathbf{22.8165 \pm 3.7317}$	$\mathbf{0.7772 \pm 0.1094}$	$\mathbf{0.0221 \pm 0.0375}$	$\mathbf{19.0374 \pm 4.1582}$
[AAAAAAB]	0.0122 ± 0.0062	19.7162 ± 2.3813	0.5947 ± 0.1913	0.0272 ± 0.0402	17.9227 ± 4.1764
[AAAAABB]	0.0318 ± 0.0216	15.8378 ± 2.7313	0.4635 ± 0.1752	0.0584 ± 0.0774	14.8665 ± 4.7720
[AAAABBB]	0.0734 ± 0.0344	11.9599 ± 2.5316	0.3481 ± 0.1775	0.1300 ± 0.1478	11.3327 ± 5.0234
[AAABBBB]	0.0820 ± 0.0385	11.4885 ± 2.5578	0.3384 ± 0.1776	0.1800 ± 0.1765	9.9064 ± 5.2330
[AABBBBB]	0.0863 ± 0.0400	11.2936 ± 2.6455	0.3300 ± 0.1897	0.1705 ± 0.1496	9.6293 ± 4.6053
[ABBBBBB]	0.0969 ± 0.0378	10.6230 ± 2.3366	0.3083 ± 0.1755	0.1796 ± 0.1473	9.1598 ± 4.3589
[BBBBBBB]	0.0921 ± 0.0406	10.9644 ± 2.5820	0.3257 ± 0.1812	0.1871 ± 0.1677	9.0777 ± 4.4095

Table 6.2. Quantitative results from the Skip Connections Analysis on MI-P2P$_{T_{2flair}}$: skips are sequentially perturbed.

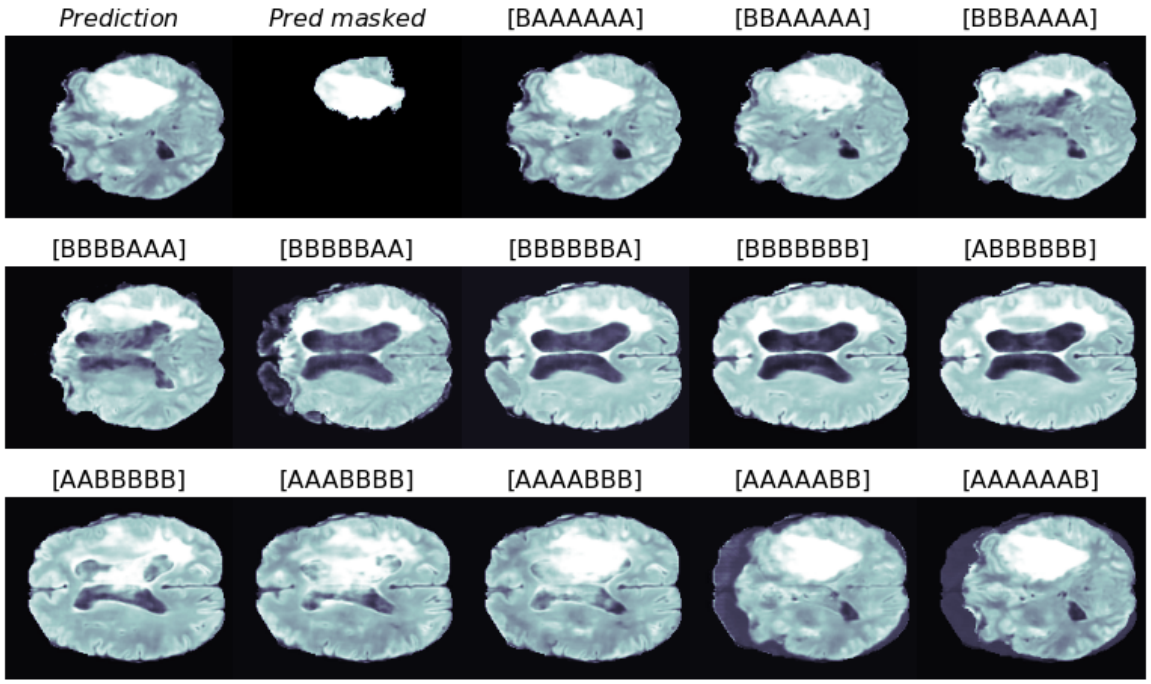


Figure 6.3. Qualitative results from the Skip Connections Analysis on MI-P2P$_{T_{2flair}}$: skips are sequentially perturbed. *Prediction* corresponds to config.[AAAAAAA]

6.2 Internal Connections Analysis

In this section we present the analysis obtained by sequentially turning off the internal connections (or channels) of the generator network. The results are reported in Table 6.3 and Figure 6.5, where, for example, a [1,1,1,1,1,0,0] configuration corresponds to switching off the 6th and 7th (and, as a consequence, the 9th and 10th, since the connections are switched in pairs) internal channels (Figure 6.4).

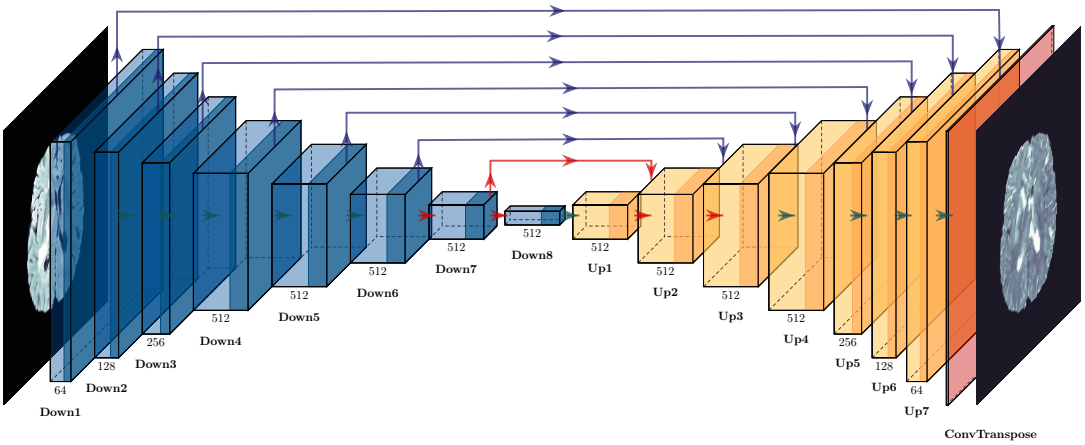


Figure 6.4. Example of network corresponding to the configuration [1,1,1,1,1,0,0] in the Internal Connections Analysis. In red the connections turned off.

It is worth to highlight that, even though this analysis focuses on the internal connections, when a pair of inner channels are off (except in the case of the 7th and 9th links), then also the entries of a skip connection are forced to 0, as it possible to observe in Figure 6.4 where all the connections turned off are shown in red color.

Internal off	**MSE**	**PSNR**	**SSIM**	**MSE$_{\text{tumor}}$**	**PSNR$_{\text{tumor}}$**
[1,1,1,1,1,1,1]	$\mathbf{0.0069 \pm 0.0049}$	$\mathbf{22.8165 \pm 3.7317}$	$\mathbf{0.7772 \pm 0.1094}$	$\mathbf{0.0221 \pm 0.0375}$	$\mathbf{19.0374 \pm 4.1582}$
[0,1,1,1,1,1,1]	0.0103 ± 0.0058	20.8939 ± 3.7816	0.7062 ± 0.1355	0.0406 ± 0.0173	14.3578 ± 2.0877
...	...	...	...	...	...
[0,0,0,0,0,0,0]	0.0103 ± 0.0058	20.8939 ± 3.7816	0.7062 ± 0.1355	0.0406 ± 0.0173	14.3578 ± 2.0877
[1,1,1,1,1,1,1]	$\mathbf{0.0069 \pm 0.0049}$	22.8165 ± 3.7317	0.7772 ± 0.1094	0.0221 ± 0.0375	$\mathbf{19.0374 \pm 4.1582}$
[1,1,1,1,1,1,0]	$\mathbf{0.0069 \pm 0.0049}$	$\mathbf{22.8750 \pm 3.7301}$	$\mathbf{0.7794 \pm 0.1084}$	$\mathbf{0.0218 \pm 0.0346}$	18.9272 ± 4.0549
[1,1,1,1,1,0,0]	0.0077 ± 0.0048	22.2194 ± 3.6022	0.7579 ± 0.1165	0.0229 ± 0.0384	18.4526 ± 3.8248
[1,1,1,1,0,0,0]	0.0095 ± 0.0054	21.2806 ± 3.7368	0.7505 ± 0.1164	0.0393 ± 0.0253	14.6543 ± 2.3238
[1,1,1,0,0,0,0]	0.0104 ± 0.0062	20.8601 ± 3.6599	0.7192 ± 0.1326	0.0400 ± 0.0454	15.1987 ± 3.1458
[1,1,0,0,0,0,0]	0.0108 ± 0.0059	20.6812 ± 3.7336	0.7403 ± 0.1178	0.0444 ± 0.0259	14.0582 ± 2.2371
[1,0,0,0,0,0,0]	0.0162 ± 0.0086	18.8502 ± 3.5119	0.6641 ± 0.1233	0.0845 ± 0.0249	10.9653 ± 1.7048
[0,0,0,0,0,0,0]	0.0103 ± 0.0058	20.8939 ± 3.7816	0.7062 ± 0.1355	0.0406 ± 0.0173	14.3578 ± 2.0877

Table 6.3. Quantitative results from the Internal Connections Analysis on MI-P2P$_{T_{2flair}}$: channels are sequentially turned off.

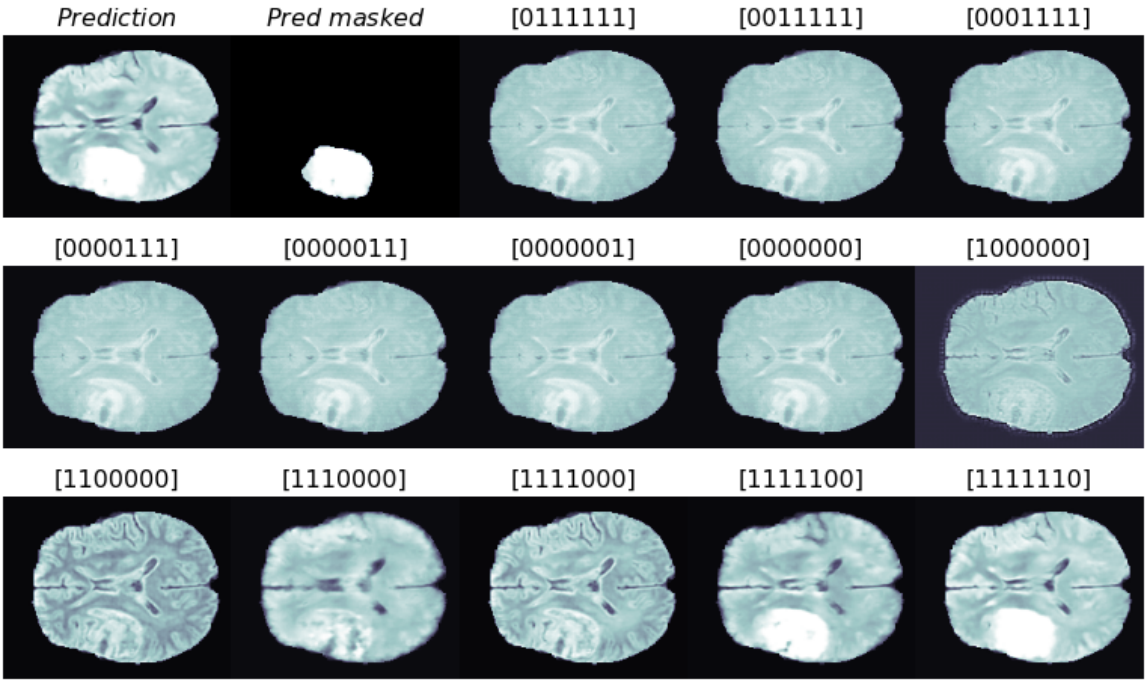


Figure 6.5. Qualitative results from the Internal Connections Analysis on MI-P2P$_{T_{2flair}}$: channels are sequentially turned off. *Prediction* corresponds to config.[1111111]

6.3 Experiments Discussion

While in Subsection 5.3.1 we demonstrated that the input received by our models isn't just outputted but it is elaborated, allowing to produce a completely new synthesized image, in Sections 6.1 and 6.2 we investigated about which connections, in a generator network, are used the most in this processing: in this section we discuss the results obtained from the experiments presented in the previous sections, eventually referring to the additional results reported in Appendix A.

Table 6.3, reporting the quantitative results from the Internal Connection Analysis, shows that all the configurations with the the first (and the last, since they are turned off in pairs) internal channels off reach the exactly same performances in the evaluation metrics. This is due to the fact that all the entries of the connection between the last and the second-last layers are set to 0 and so the information coming from the input is sent to the output only through the outermost skip connection.

Looking at the values from Table 6.3 it is possible to appreciate how the values in the performances don't deteriorate too much compared to the configuration with all the channels on. This is a first indicator about the importance of the skips connections in the generator networks: with all the internal connections off and with all the skips off with the exception of the outermost, the network is able to still obtain satisfiable results, with a worsening in the performance of 49% in MSE, 8% in PSNR, 9% in SSIM, 84% in MSE_{tumor} and 25% in $PSNR_{tumor}$.

Observing Tables 6.1 and 6.2 from the Skip Connection Analysis, we can confirm the major role played by the outermost skip connection in the generative process: when all the skips values are forced to 0 and the information passes only through internal connections, the network performs worse, compared to the [1111111] configuration, by a significant degradation of 1330% in MSE, 55% in PSNR, 97% in SSIM, 1362% in MSE_{tumor} and 73% in $PSNR_{tumor}$ (Figure 6.6).

This degradation in the performances can be easily appreciated also by a visual comparison of the qualitative results shown in Figure 6.2 and it reinforces the hypothesis that these connections are crucial for a generative model since they allow to recover the information lost in the contracting path through the many downsampling blocks.

The outermost skip is for sure the connection that brings the most significant source of information to the output image but in general, looking both at Figure 6.2 and Table 6.1, we can observe how the scores obtained from the metrics drop to

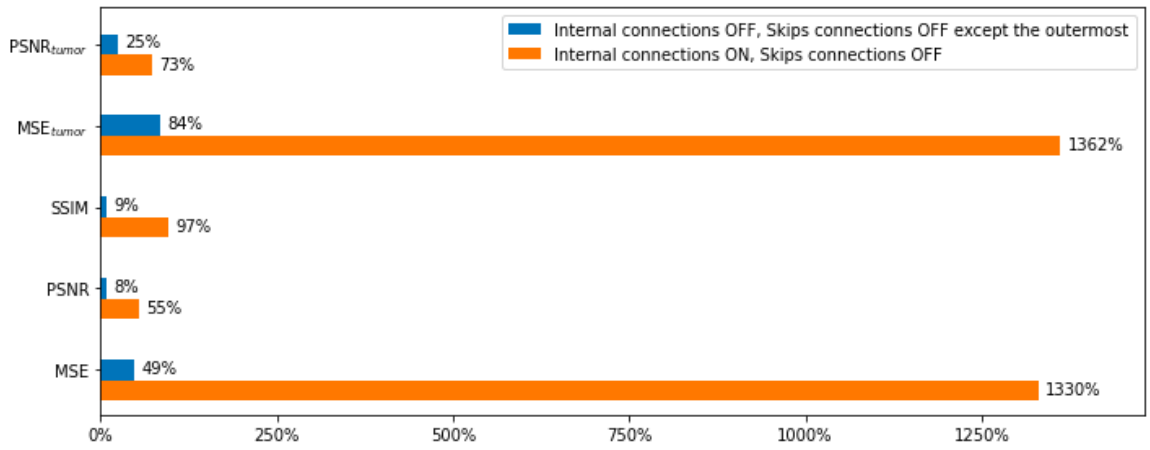


Figure 6.6. Percentual degradation observed in the performances of MI-pix2pix, with respect to the configuration with all the connections, skips and internal, ON (Config. [1111111]).

low values more rapidly by switching off the outer connections than switching first the inner ones, due to the fact the most external skips are crucial for a Generative Adversarial network.

The results we obtained in the second part (skips perturbed, Subsection 6.1.2) of the Skip Connections Analysis are consistent with what was observed in the first one (skips off, Subsection 6.1.1) and show a similar behaviour in the degradation of the performances. The qualitative results, instead, show even better what we said about the most external links: Figure 6.3 illustrates how the appearance belonging to the image A is completely almost compromised after perturbing the generator with the replacement, in two outermost skip connections, of the content belonging to A with the one produced by another network that takes as input B.

Further experiments on MI-GAN and pix2pix, that can be found in Appendix A, show that the unimodal model, compared to two multi-modal networks, seems to have more balance in the gap between the contributions of the skip connections to the generated image: the outermost skip connections remain the most significant for the generative process but to a lesser extent with respect to what we observed with the multi-input generator.

This can be visually appreciated by a comparison between Figures A.1 and 6.2 but also by the quantitative results from the analysis on the internal connections, with pix2pix that, switching off all the connections except for the last skip channel, deteriorates more (238% in MSE, 25% in PSNR, 21% in SSIM) than MI-pix2pix (49% in MSE, 8% in PSNR, 9% in SSIM) and MI-GAN (72% in MSE, 11% in PSNR, 15% in SSIM) (Figure 6.7).

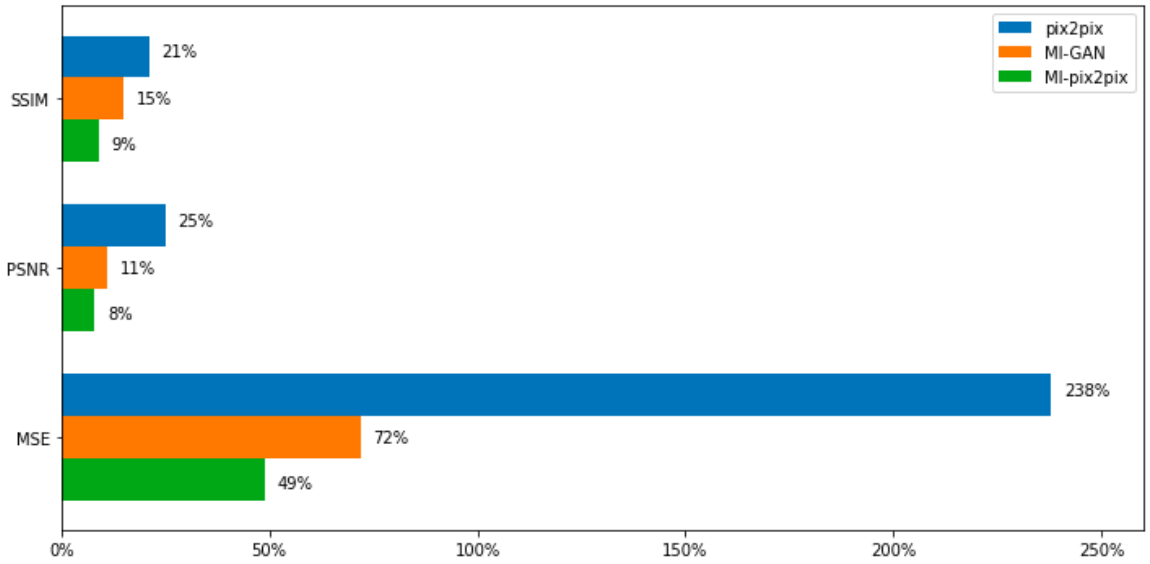


Figure 6.7. Percentual degradation observed in the performances of the different models when all the internal channels are turned off (and so every skip connections is off, except for the outermost one), with respect to the configuration with all the connections, skips and internal, ON (Config. [1111111]).

6.4 Summary

In this chapter we presented the experiments conducted in order to investigate about which are the connections used the most during the generative process of the models we trained.

Then we reported and evaluated both the quantitative and the qualitative results reached with our tests, discussing the observed behaviour of the generator network and showing the insights provided by the Skip Connections Analysis and by the Internal Connections Analysis: the experiments performed led us to the conclusion that skip connections (in particular the outermost skip connections) have a crucial role in the generative process and by turning off or perturbing these channels, a significant degradation in the performances of the network is observed.

In the next chapter, we provide more general considerations on our work and we present the open problems as well as the possible future works.

Chapter 7

Conclusions and Future Work

In this chapter we make our final considerations about this work: first by reporting our conclusions in Section 7.1, then by discussing the open problems and the possible applications in which the framework presented could be used (Section 7.2).

7.1 Conclusions

In Chapter 5 we showed the results obtained by using Generative Adversarial Networks to synthesize missing modalities after training the models with thousands of brain MRI slices: the generated images are highly accurate and realistic, compared to sequences belonging to the dataset, and in many cases it is objectively difficult to distinguish between true and fake images.

Because of this we are overall satisfied by the generation quality of the images and by the achieved results (measured by the different evaluation metrics implemented) that are extremely encouraging for future develops and demonstrate the applicability of GANs in neuroimaging. In Chapter 6, then, we also showed, through various experiments on the trained models, the effectiveness of the skip connections, crucial to recover spatial information otherwise lost in the downsampling part of the Generator network.

We found cases where the image wasn't a realistic copy of the ground truth, but not due to some bad implementation choices but because the missing modality generated is originally obtained using a contrast agent and so it has a highly specific information difficult to reproduce and not present in any other sequence.

However, the models implemented are far from being perfect with the generated sequences that in some cases contained blurred artifacts or low-level details not perfectly defined.

In general, the multi-modal models tested have proved to be better than unimodal GANs (except in the case in which target and input images present highly similar characteristics), due to the fact that they can exploit the correlation between available sequences, allowing us to reduce the order of magnitude of required models (from 12 single-input single-output to 4 multi-input single-output). The results obtained in this work demonstrate the possibility of using successfully this generative framework as a part of a bigger pipeline that can generate and then segment the synthesized missing modality, giving to the doctors an additional aid for the diagnosis of brain tumors.

7.2 Future Work

In this section we describe the open problems encountered during the work and we propose different solutions to improve the system designed and to make possible, in the future, its employment in medical imaging.

7.2.1 Open Problems

Data Availability

The amount of data available is crucial for any deep learning setting and the scarcity of annotated samples represents a huge problem, especially, for all the systems working in the medical imaging domain. In our case, the first problem we had to deal with, working with a medical dataset, was that BRATS2015, built for a segmentation challenge, was composed by two parts: one intended for the training and one for the testing, without annotated and segmented samples. So we could use only the first set of data, since we needed ground truths to evaluate the tumor area and to assess the quality of the segmentation obtained with the GAN predictions.

The second problem we faced was that our reduced dataset is composed by 35.072 slices per modality (with 28.160 used for the training) but all these images belong to the volumes of 274 subjects (training set reduced to 219 patients), so the samples in the dataset are not independent and most of these slices contain very similar information.

Evaluation Metrics

Another common and open problem is the one related to evaluation of the results produced by a Generative Adversarial Network: many works, and so did we, adopt metrics such as PSNR and SSIM to quantitatively evaluate the generated image but the problem is that these measures don't correspond to the visual quality of the image.

GAN evaluation is not a settled issue yet and researchers suggest to validate the quality of the prediction by using segmentation models, as we did in Subsection 5.2.2, or, even better, to recruit domain experts that could evaluate the results generated by the network: this, unfortunately, is not always accomplishable since it is expensive and time-consuming [2].

7.2.2 Possible Applications and Future Develops

Improving the MSE and PSNR implemented

In our work we calculated the MSE and PSNR over the whole image, considering the pixels belonging to the brain as well as the ones with zero value. The reasoning behind is that the GAN learns to generate also the black pixels so it is not wrong to measure the similarity and the error over the overall area.

An alternative approach could be to modify the MSE and the PSNR implementation considering only the pixels with value different from zero and the ones not belonging to the tumor: by doing this it would be possible to understand, through a direct comparison, if the generation quality in the malignant area is higher/equal/lower with respect to the rest of the brain.

Tweaking the model

In our case we used the hyperparameters values defined by [6] and [1], since we couldn't find any improvements over the defined values. We believe that networks capabilities can be increased by tweaking the model to find the optimal parameters.

Variable number of inputs to the multi-modal model

One way to improve the multi-modal networks developed would be to train them to synthesize one image in the presence of a variable numbers of input sequences (while in our case, we need to use the slices from all the three different modalities, in order to be able to synthesize the missing image).

Multi-input Multi-output model

It would be interesting, then, to study possible improvements in the direction of multi-input multi-output models, where some researchers have already started to experiment with, such as Sharma et al. with their proposed MM-GAN [1].

Possible Applications

The most obvious application of our work, as discussed in previous sections, would be to use the developed network as a part of a bigger pipeline of downstream analysis, such as segmentation, in order to improve the daily workflow of the radiologists. Some segmentation models indeed depend on the implicit assumption that all the input sequences are available: because of this, if some modalities, due to various reasons, are missing, then a generative adversarial network trained on brain MRI would be very useful to synthesize the images needed.

This generative framework has many possible applications in different medical settings and lots of studies have been focusing on how to get the most from these networks. For instance, GANs would be very useful in MRI image restoration where artifacts due to errors/motions during the scan can be removed, avoiding to repeat the exams multiple times. However, it's important to underline that, as Yi et al. suggests in [2], GAN-based methods are still far from being adopted clinically: this field is still in its infancy and a lot of work has to be done yet.

Acronyms

AI	Artificial Intelligence
cGAN	conditional GAN
CNN	Convolutional Neural Network
CSF	Cerebrospinal fluid
CT	Computerized Tomography
DL	Deep Learning
GAN	Generative Adversarial Network
GPU	Graphics processing unit
MR	Magnetic Resonance
MRA	Magnetic Resonance Angiohraphy
MRI	Magnetic Resonance Imaging
PET	Positron Emission Tomography

Bibliography

[1] A. Sharma and G. Hamarneh, "Missing mri pulse sequence synthesis using multi-modal generative adversarial network," 2019. [Online]. Available: https://arxiv.org/abs/1904.12200

[2] X. Yi, E. Walia, and P. Babyn, "Generative adversarial network in medical imaging: A review," *Medical Image Analysis*, vol. 58, p. 101552, Dec 2019. [Online]. Available: http://dx.doi.org/10.1016/j.media.2019.101552

[3] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, p. 60–88, Dec 2017. [Online]. Available: http://dx.doi.org/10.1016/j.media.2017.07.005

[4] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, p. 1097–1105, 2012. [Online]. Available: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[6] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," 2016. [Online]. Available: http://arxiv.org/abs/1611.07004

[7] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE 86(11)*, p. 2278–2324, 1998. [Online]. Available: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

[8] J. M. Susskind, A. K. Anderson, and G. Hinton, "The toronto face dataset. technical report utml tr 2010-001," 2010.

[9] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images. technical report," 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[10] M. Orbes-Arteaga, M. J. Cardoso, L. Sørensen, M. Modat, S. Ourselin, M. Nielsen, and A. Pai, "Simultaneous synthesis of flair and segmentation of white matter hypointensities from t1 mris," 2018. [Online]. Available: https://arxiv.org/abs/1808.06519

[11] F. Calimeri, A. Marzullo, C. Stamile, and G. Terracina, "Biomedical data augmentation using generative adversarial neural networks," in *Artificial Neural Networks and Machine Learning – ICANN 2017*, 10 2017, pp. 626–634.

[12] S. U. H. Dar, M. Yurt, L. Karacan, A. Erdem, E. Erdem, and T. Çukur, "Image synthesis in multi-contrast mri with conditional generative adversarial networks," 2018. [Online]. Available: https://arxiv.org/abs/1802.01221

[13] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2017. [Online]. Available: https://arxiv.org/abs/1703.10593

[14] S. Olut, Y. H. Sahin, U. Demir, and G. Unal, "Generative adversarial training for mra image synthesis using multi-contrast mri," 2018. [Online]. Available: https://arxiv.org/abs/1804.04366

[15] A. Ben-Cohen, E. Klang, S. P. Raskin, S. Soffer, S. Ben-Haim, E. Konen, M. M. Amitai, and H. Greenspan, "Cross-modality synthesis from ct to pet using fcn and gan networks for improved automated lesion detection," 2018. [Online]. Available: https://arxiv.org/abs/1802.07846

[16] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014. [Online]. Available: http://arxiv.org/abs/1411.1784

[17] T. Silva, "An intuitive introduction to generative adversarial networks (gans)," 01 2018. [Online]. Available: https://www.freecodecamp.org/news/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394/

[18] C. Floudas and P. Pardalos, *Encyclopedia of Optimization*, ser. Encyclopedia of Optimization. Kluwer Academic, 2001, no. v. 1. [Online]. Available: https://books.google.com.mx/books?id=gtoTkL7heS0C

[19] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2017. [Online]. Available: https://books.google.com.mx/books?id=bRpYDgAAQBAJ

[20] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015. [Online]. Available: http://arxiv.org/abs/1511.06434

[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015. [Online]. Available: https://arxiv.org/abs/1502.03167

[22] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010. [Online]. Available: https://www.cs.toronto.edu/~fritz/absps/reluICML.pdf

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. [Online]. Available: https://arxiv.org/abs/1505.04597

[24] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," 2016. [Online]. Available: https://arxiv.org/abs/1604.04382

[25] "Magnetic resonance imaging (mri)," Winship Cancer Institute of Emory University. [Online]. Available: https://www.cancerquest.org/patients/detection-and-diagnosis/magnetic-resonance-imaging-mri

[26] K. K. Farmanfarma, M. Mohammadian, Z. Shahabinia, S. Hassanipour, and H. Salehiniya, "Brain cancer in the world: an epidemiological review," *WCRJ 2019*, no. 6: e1356, 07 2019. [Online]. Available: https://www.wcrj.net/wp-content/uploads/sites/5/2019/07/e1356-Brain-cancer-in-the-world-an-epidemiological-review.pdf

[27] B. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, L. Lanczi, E. Gerstner, M.-A. Weber, T. Arbel, B. Avants, N. Ayache, P. Buendia, L. Collins, N. Cordier, J. Corso, A. Criminisi, T. Das, H. Delingette, C. Demiralp, C. Durst, M. Dojat, S. Doyle, J. Festa, F. Forbes, E. Geremia, B. Glocker, P. Golland, X. Guo, A. Hamamci, K. Iftekharuddin, R. Jena, N. John, E. Konukoglu, D. Lashkari,

J. Antonio Mariz, R. Meier, S. Pereira, D. Precup, S. J. Price, T. Riklin-Raviv, S. Reza, M. Ryan, L. Schwartz, H.-C. Shin, J. Shotton, C. Silva, N. Sousa, N. Subbanna, G. Szekely, T. Taylor, O. Thomas, N. Tustison, G. Unal, F. Vasseur, M. Wintermark, D. Hye Ye, L. Zhao, B. Zhao, D. Zikic, M. Prastawa, M. Reyes, and K. Van Leemput, "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)," *IEEE Transactions on Medical Imaging*, p. 33, 2014. [Online]. Available: https://hal.inria.fr/hal-00935640

[28] M. Kistler, S. Bonaretti, M. Pfahrer, R. Niklaus, and P. Büchler, "The virtual skeleton database: An open access repository for biomedical research and collaboration," *J Med Internet Res*, vol. 15, no. 11, p. e245, Nov 2013. [Online]. Available: http://www.jmir.org/2013/11/e245/

[29] D. Reardon, "What makes a brain tumor high-grade or low-grade?" Dana-Farber Cancer Institute. [Online]. Available: https://blog.dana-farber.org/insight/2018/04/makes-brain-tumor-high-grade-low-grade/

[30] "Tumor grade," NIH National cancer Institute. [Online]. Available: https://www.cancer.gov/about-cancer/diagnosis-staging/prognosis/tumor-grade-fact-sheet

[31] Y. Xue, T. Xu, H. Zhang, L. R. Long, and X. Huang, "Segan: Adversarial network with multi-scale l1 loss for medical image segmentation," *Neuroinformatics*, vol. 16, no. 3-4, p. 383–392, May 2018. [Online]. Available: http://dx.doi.org/10.1007/s12021-018-9377-x

[32] "Tfrecord and tf.example." [Online]. Available: https://www.tensorflow.org/tutorials/load_data/tfrecord

[33] "Pix2pix." [Online]. Available: https://www.tensorflow.org/tutorials/generative/pix2pix

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[35] A. Ng, "Gradient descent in practice I - feature scaling," Coursera. [Online]. Available: https://www.coursera.org/learn/machine-learning/lecture/xx3Da/gradient-descent-in-practice-i-feature-scaling

[36] "Psnr," Mathworks. [Online]. Available: https://it.mathworks.com/help/vision/ref/psnr.html

[37] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *Image Processing, IEEE Transactions on*, vol. 13, pp. 600 – 612, 05 2004. [Online]. Available: https://ieeexplore.ieee.org/document/1284395

[38] E. Giacomello, D. Loiacono, and L. Mainardi, "Brain mri tumor segmentation with adversarial networks," 2019. [Online]. Available: https://arxiv.org/abs/1910.02717

Appendix A

Appendix: Further Results from Connections Analysis

In this Appendix we report additional (quantitative and qualitative) results obtained from the following experiments (in the same order as they are presented):

- Skip Connections Analysis on P2P($T_{2\rightarrow 2flair}$) and MI-GAN$_{T_{2flair}}$ with channels turned off.
- Skip Connections Analysis on P2P($T_{2\rightarrow 2flair}$) and MI-GAN$_{T_{2flair}}$ with channels perturbed.
- Internal Connections Analysis on P2P($T_{2\rightarrow 2flair}$) and MI-GAN$_{T_{2flair}}$ with channels turned off.

In the following tables, values in boldface represent best performance values. The reported values are mean $\pm$ std.

Table A.1. Quantitative results from the Skip Connections Analysis on P2P($T_{2\to 2flair}$): skips are sequentially turned off.

Skips off	**MSE**	**PSNR**	**SSIM**	**MSE$_{\text{tumor}}$**	**PSNR$_{\text{tumor}}$**
[1,1,1,1,1,1,1]	0.0090 ± 0.0065	$\mathbf{21.5895 \pm 3.4831}$	$\mathbf{0.7518 \pm 0.1211}$	$\mathbf{0.0390 \pm 0.0463}$	15.9946 ± 4.0459
[0,1,1,1,1,1,1]	$\mathbf{0.0090 \pm 0.0062}$	21.5423 ± 3.3860	0.7515 ± 0.1204	0.0391 ± 0.0473	$\mathbf{16.0047 \pm 4.0874}$
[0,0,1,1,1,1,1]	0.0099 ± 0.0068	21.1858 ± 3.6324	0.7436 ± 0.1268	0.0464 ± 0.0422	14.6891 ± 3.6658
[0,0,0,1,1,1,1]	0.0139 ± 0.0086	19.7127 ± 3.7077	0.7069 ± 0.1398	0.0792 ± 0.0634	12.0341 ± 3.1018
[0,0,0,0,1,1,1]	0.0147 ± 0.0086	19.4145 ± 3.5775	0.6591 ± 0.1253	0.0845 ± 0.0370	11.1662 ± 2.0567
[0,0,0,0,0,1,1]	0.0160 ± 0.0104	19.0200 ± 3.4598	0.6440 ± 0.1113	0.0440 ± 0.0255	14.2132 ± 2.3999
[0,0,0,0,0,0,1]	0.0494 ± 0.0285	13.8789 ± 2.8530	0.1962 ± 0.0692	0.1423 ± 0.0613	8.8606 ± 1.9105
[0,0,0,0,0,0,0]	0.1229 ± 0.0630	9.7545 ± 2.5459	-0.0072 ± 0.0352	0.3883 ± 0.1420	4.3978 ± 1.6224
[1,1,1,1,1,1,1]	$\mathbf{0.0090 \pm 0.0065}$	$\mathbf{21.5895 \pm 3.4831}$	$\mathbf{0.7518 \pm 0.1211}$	$\mathbf{0.0390 \pm 0.0463}$	$\mathbf{15.9946 \pm 4.0459}$
[1,1,1,1,1,1,0]	0.0155 ± 0.0085	18.8593 ± 2.9462	0.6954 ± 0.1149	0.0682 ± 0.0726	13.1789 ± 3.6046
[1,1,1,1,1,0,0]	0.0431 ± 0.0194	14.1364 ± 2.1434	0.3764 ± 0.1560	0.1378 ± 0.1250	9.9799 ± 3.6409
[1,1,1,1,0,0,0]	0.0380 ± 0.0187	14.7498 ± 2.2690	0.3761 ± 0.1271	0.0915 ± 0.0927	11.9744 ± 3.7576
[1,1,1,0,0,0,0]	0.0789 ± 0.0286	11.3898 ± 1.9330	0.1219 ± 0.0759	0.1838 ± 0.1397	8.3547 ± 3.0633
[1,1,0,0,0,0,0]	0.0982 ± 0.0319	10.3641 ± 1.7303	0.0404 ± 0.0606	0.2792 ± 0.1765	6.1384 ± 2.1748
[1,0,0,0,0,0,0]	0.0929 ± 0.0335	10.5920 ± 1.5696	0.0489 ± 0.0610	0.2497 ± 0.1228	6.4492 ± 1.8823
[0,0,0,0,0,0,0]	0.1229 ± 0.0630	9.7545 ± 2.5459	-0.0072 ± 0.0352	0.3883 ± 0.1420	4.3978 ± 1.6224

Figure A.1. Qualitative results from the Skip Connections Analysis on P2P($T_{2\to 2flair}$): skips are sequentially turned off. *Prediction* corresponds to config.[1111111]

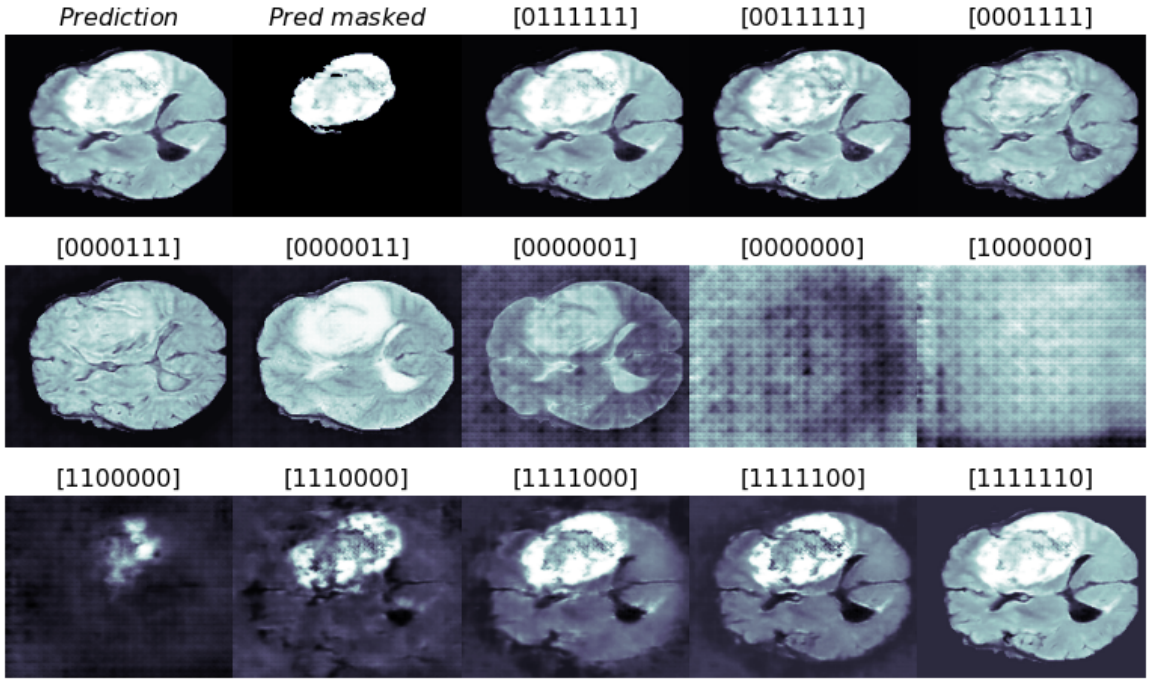


Table A.2. Quantitative results from the Skip Connections Analysis on MI-GAN$_{T_{2flair}}$: skips are sequentially turned off.

Skips off	**MSE**	**PSNR**	**SSIM**	**MSE$_{\text{tumor}}$**	**PSNR$_{\text{tumor}}$**
[1,1,1,1,1,1,1]	$\mathbf{0.0072 \pm 0.0050}$	$\mathbf{22.5524 \pm 3.5655}$	$\mathbf{0.7610 \pm 0.1175}$	$\mathbf{0.0258 \pm 0.0285}$	$\mathbf{17.4694 \pm 3.6137}$
[0,1,1,1,1,1,1]	0.0081 ± 0.0051	21.9370 ± 3.3825	0.7439 ± 0.1210	0.0277 ± 0.0305	17.1080 ± 3.5332
[0,0,1,1,1,1,1]	0.0100 ± 0.0065	21.0970 ± 3.6387	0.7375 ± 0.1233	0.0346 ± 0.0165	15.1511 ± 2.3996
[0,0,0,1,1,1,1]	0.0138 ± 0.0069	19.3884 ± 3.1246	0.6734 ± 0.1387	0.0496 ± 0.0448	14.2094 ± 3.3076
[0,0,0,0,1,1,1]	0.0109 ± 0.0074	20.8007 ± 3.7417	0.7324 ± 0.1244	0.0331 ± 0.0157	15.3792 ± 2.5432
[0,0,0,0,0,1,1]	0.0125 ± 0.0072	19.8870 ± 2.9944	0.5906 ± 0.0975	0.0389 ± 0.0172	14.5971 ± 2.2820
[0,0,0,0,0,0,1]	0.0273 ± 0.0109	16.0275 ± 1.8843	0.1886 ± 0.0422	0.1012 ± 0.0370	10.2183 ± 1.5814
[0,0,0,0,0,0,0]	0.1294 ± 0.0434	9.2003 ± 1.8129	0.0039 ± 0.0061	0.3872 ± 0.1252	4.3380 ± 1.3904
[1,1,1,1,1,1,1]	$\mathbf{0.0072 \pm 0.0050}$	$\mathbf{22.5524 \pm 3.5655}$	$\mathbf{0.7610 \pm 0.1175}$	$\mathbf{0.0258 \pm 0.0285}$	$\mathbf{17.4694 \pm 3.6137}$
[1,1,1,1,1,1,0]	0.0077 ± 0.0051	22.1638 ± 3.3988	0.7566 ± 0.1170	0.0301 ± 0.0335	16.7878 ± 3.5362
[1,1,1,1,1,0,0]	0.0424 ± 0.0213	14.5579 ± 3.3720	0.4903 ± 0.1508	0.1013 ± 0.0885	11.2495 ± 3.3876
[1,1,1,1,0,0,0]	0.0886 ± 0.0293	10.8085 ± 1.6658	0.0805 ± 0.0646	0.2480 ± 0.1308	6.4845 ± 1.8742
[1,1,1,0,0,0,0]	0.1157 ± 0.0352	9.6557 ± 1.8085	-0.0183 ± 0.0419	0.3456 ± 0.1628	4.9657 ± 1.6754
[1,1,0,0,0,0,0]	0.1266 ± 0.0397	9.2731 ± 1.8176	-0.0275 ± 0.0328	0.3742 ± 0.1696	4.6018 ± 1.6302
[1,0,0,0,0,0,0]	0.1176 ± 0.0398	9.6350 ± 1.9156	0.0060 ± 0.0162	0.3659 ± 0.1532	4.6860 ± 1.6421
[0,0,0,0,0,0,0]	0.1294 ± 0.0434	9.2003 ± 1.8129	0.0039 ± 0.0061	0.3872 ± 0.1252	4.3380 ± 1.3904

Figure A.2. Qualitative results from the Skip Connections Analysis on MI-GAN$_{T_{2flair}}$: skips are sequentially turned off. *Prediction* corresponds to config.[1111111]

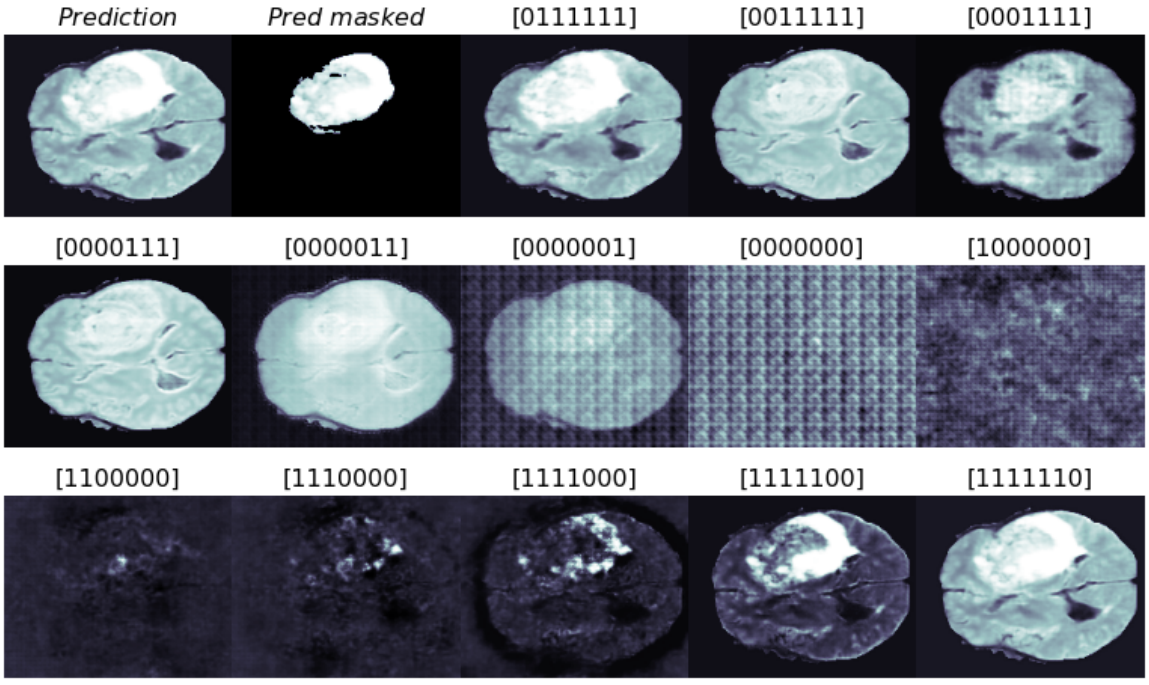


Table A.3. Quantitative results from the Skip Connections Analysis on P2P($T_{2\to 2flair}$): skips are sequentially perturbed.

Img.in Skips	**MSE**	**PSNR**	**SSIM**	**MSE$_{\text{tumor}}$**	**PSNR$_{\text{tumor}}$**
[AAAAAAA]	$\mathbf{0.0090 \pm 0.0065}$	$\mathbf{21.5895 \pm 3.4831}$	$\mathbf{0.7518 \pm 0.1211}$	$\mathbf{0.0390 \pm 0.0463}$	$\mathbf{15.9946 \pm 4.0459}$
[BAAAAAA]	0.0091 ± 0.0064	21.5235 ± 3.4479	0.7513 ± 0.1210	0.0396 ± 0.0460	15.8619 ± 3.9731
[BBAAAAA]	0.0104 ± 0.0069	20.8839 ± 3.4477	0.7383 ± 0.1270	0.0505 ± 0.0435	14.2265 ± 3.4858
[BBBAAAA]	0.0158 ± 0.0087	18.8661 ± 3.0820	0.6812 ± 0.1324	0.0802 ± 0.0542	11.9409 ± 3.2647
[BBBBAAA]	0.0271 ± 0.0137	16.3542 ± 2.7462	0.5572 ± 0.1547	0.0869 ± 0.0610	11.6530 ± 3.2698
[BBBBBAA]	0.0315 ± 0.0140	15.5757 ± 2.5717	0.5172 ± 0.1395	0.0774 ± 0.0680	12.4861 ± 3.6620
[BBBBBBA]	0.0652 ± 0.0291	12.5499 ± 2.9432	0.3767 ± 0.1851	0.1536 ± 0.1036	9.1973 ± 3.3652
[BBBBBBB]	0.0889 ± 0.0391	11.1201 ± 2.5996	0.3243 ± 0.1796	0.1904 ± 0.1628	8.8932 ± 4.3094
[AAAAAAA]	$\mathbf{0.0090 \pm 0.0065}$	$\mathbf{21.5895 \pm 3.4831}$	$\mathbf{0.7518 \pm 0.1211}$	$\mathbf{0.0390 \pm 0.0463}$	$\mathbf{15.9946 \pm 4.0459}$
[AAAAAAB]	0.0143 ± 0.0073	18.9923 ± 2.2552	0.6101 ± 0.1438	0.0572 ± 0.0600	13.9484 ± 3.6776
[AAAAABB]	0.0515 ± 0.0250	13.4727 ± 2.4009	0.4279 ± 0.1666	0.1078 ± 0.1090	11.5761 ± 4.4659
[AAAABBB]	0.0494 ± 0.0199	13.4762 ± 2.0546	0.4102 ± 0.1527	0.0797 ± 0.0888	12.7056 ± 3.9738
[AAABBBB]	0.0772 ± 0.0358	11.7477 ± 2.5649	0.3429 ± 0.1745	0.1470 ± 0.1443	10.5189 ± 4.9880
[AABBBBB]	0.0849 ± 0.0384	11.3273 ± 2.5810	0.3230 ± 0.1862	0.1747 ± 0.1421	9.2217 ± 4.1810
[ABBBBBB]	0.0932 ± 0.0361	10.7810 ± 2.2974	0.3084 ± 0.1730	0.1833 ± 0.1458	8.9416 ± 4.0940
[BBBBBBB]	0.0889 ± 0.0391	11.1201 ± 2.5996	0.3243 ± 0.1796	0.1904 ± 0.1628	8.8932 ± 4.3094

Figure A.3. Qualitative results from the Skip Connections Analysis on P2P($T_{2\to 2flair}$): skips are sequentially perturbed. *Prediction* corresponds to config.[AAAAAAA]

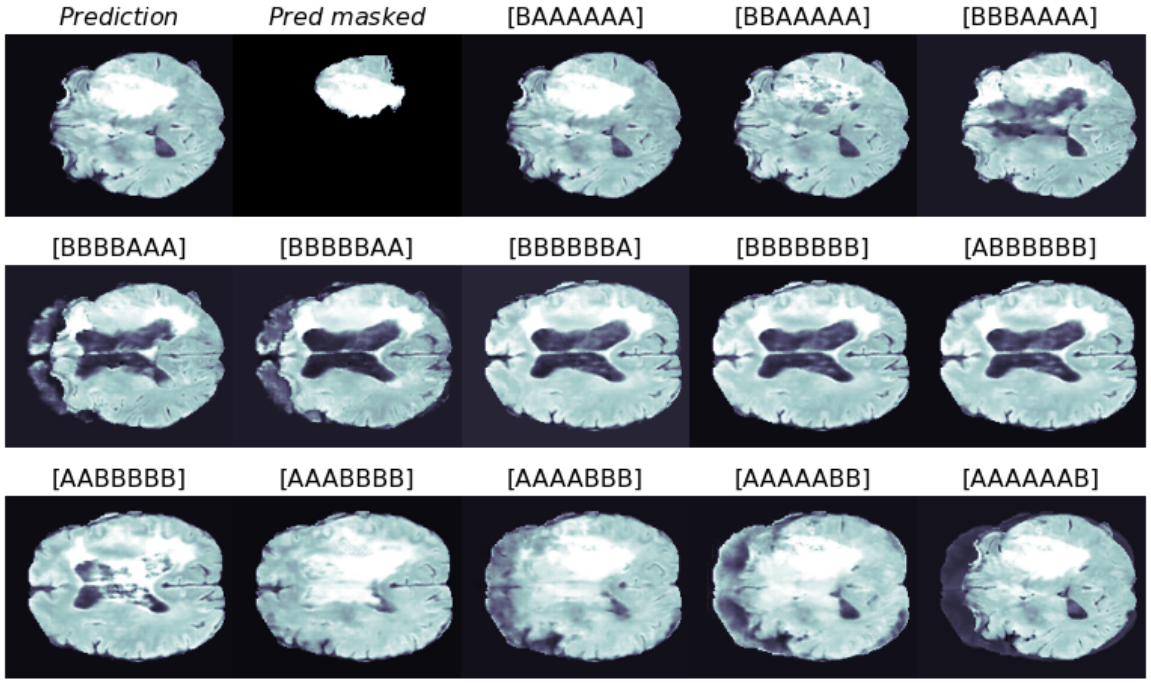


Table A.4. Quantitative results from the Skip Connections Analysis on MI-GAN$_{T_{2flair}}$: skips are sequentially perturbed.

Img.in Skips	**MSE**	**PSNR**	**SSIM**	**MSE$_{\text{tumor}}$**	**PSNR$_{\text{tumor}}$**
[AAAAAAA]	$\mathbf{0.0072 \pm 0.0050}$	$\mathbf{22.5524 \pm 3.5655}$	$\mathbf{0.7610 \pm 0.1175}$	$\mathbf{0.0258 \pm 0.0285}$	$\mathbf{17.4694 \pm 3.6137}$
[BAAAAAA]	0.0075 ± 0.0050	22.3512 ± 3.5176	0.7560 ± 0.1174	0.0262 ± 0.0282	17.2649 ± 3.4428
[BBAAAAA]	0.0092 ± 0.0057	21.3856 ± 3.5600	0.7361 ± 0.1223	0.0358 ± 0.0277	15.4185 ± 2.9900
[BBBAAAA]	0.0112 ± 0.0070	20.5974 ± 3.5783	0.7166 ± 0.1312	0.0466 ± 0.0308	14.2036 ± 3.1104
[BBBBAAA]	0.0139 ± 0.0081	19.5630 ± 3.4078	0.6884 ± 0.1308	0.0544 ± 0.0365	13.5932 ± 3.1872
[BBBBBAA]	0.0550 ± 0.0331	13.4674 ± 2.9789	0.4081 ± 0.1542	0.1441 ± 0.1447	10.7394 ± 4.9478
[BBBBBBA]	0.0779 ± 0.0359	11.8374 ± 3.0566	0.3581 ± 0.1924	0.1724 ± 0.1329	9.0690 ± 3.9221
[BBBBBBB]	0.0885 ± 0.0395	11.1499 ± 2.6073	0.3269 ± 0.1817	0.1854 ± 0.1660	9.0842 ± 4.3384
[AAAAAAA]	$\mathbf{0.0072 \pm 0.0050}$	$\mathbf{22.5524 \pm 3.5655}$	$\mathbf{0.7610 \pm 0.1175}$	$\mathbf{0.0258 \pm 0.0285}$	$\mathbf{17.4694 \pm 3.6137}$
[AAAAAAB]	0.0079 ± 0.0050	21.8535 ± 2.8257	0.7075 ± 0.1378	0.0277 ± 0.0293	17.0468 ± 3.4481
[AAAAABB]	0.0223 ± 0.0210	18.3650 ± 4.1216	0.6110 ± 0.1769	0.0603 ± 0.0783	14.8012 ± 4.8060
[AAAABBB]	0.0755 ± 0.0366	11.8979 ± 2.6665	0.3496 ± 0.1840	0.1510 ± 0.1787	11.0146 ± 5.2233
[AAABBBB]	0.0825 ± 0.0391	11.4955 ± 2.6361	0.3422 ± 0.1808	0.1824 ± 0.1783	9.8509 ± 5.2288
[AABBBBB]	0.0851 ± 0.0401	11.3747 ± 2.6967	0.3300 ± 0.1918	0.1703 ± 0.1484	9.6179 ± 4.6436
[ABBBBBB]	0.0932 ± 0.0364	10.7946 ± 2.3412	0.3106 ± 0.1764	0.1799 ± 0.1449	9.0683 ± 4.2253
[BBBBBBB]	0.0885 ± 0.0395	11.1499 ± 2.6073	0.3269 ± 0.1817	0.1854 ± 0.1660	9.0842 ± 4.3384

Figure A.4. Qualitative results from the Skip Connections Analysis on MI-GAN$_{T_{2flair}}$: skips are sequentially perturbed. *Prediction* corresponds to config.[AAAAAAA]

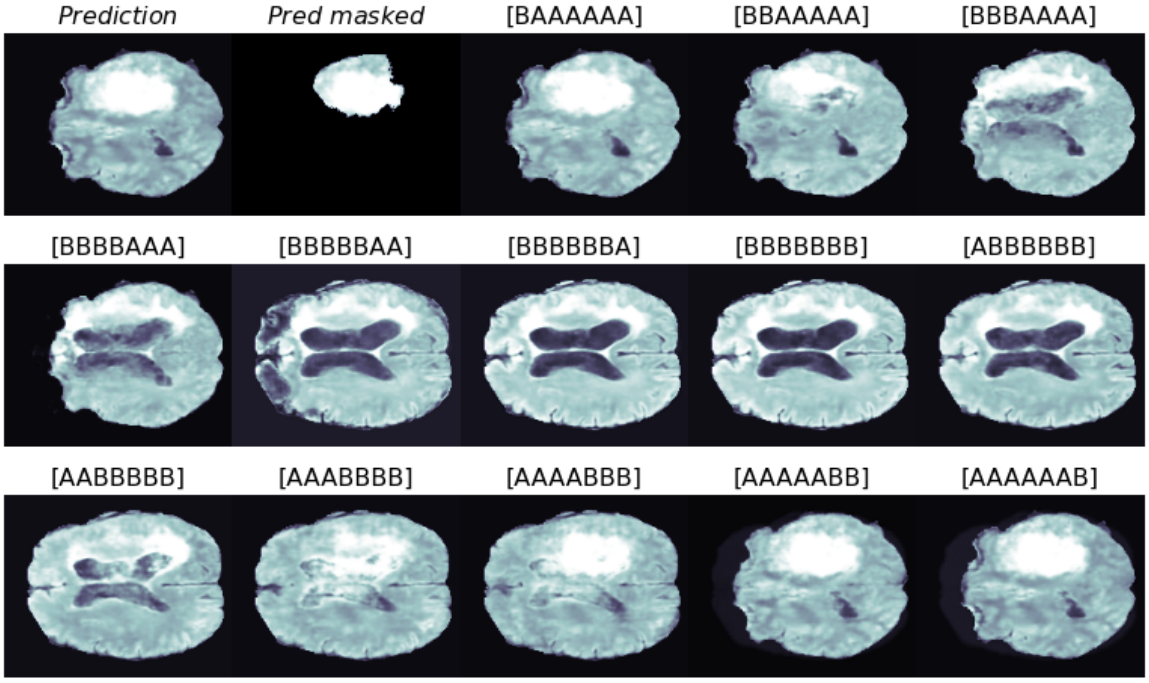


Table A.5. Quantitative results from the Internal Connections Analysis on P2P($T_{2\rightarrow 2flair}$): channels are sequentially turned off.

Internal off	**MSE**	**PSNR**	**SSIM**	**MSE$_{\text{tumor}}$**	**PSNR$_{\text{tumor}}$**
[1,1,1,1,1,1,1]	$\mathbf{0.0090 \pm 0.0065}$	$\mathbf{21.5895 \pm 3.4831}$	$\mathbf{0.7518 \pm 0.1211}$	$\mathbf{0.0390 \pm 0.0463}$	$\mathbf{15.9946 \pm 4.0459}$
[0,1,1,1,1,1,1]	0.0304 ± 0.0188	16.2855 ± 3.9058	0.5935 ± 0.1641	0.0728 ± 0.0414	12.0387 ± 2.4420
...	...	...	...	...	...
[0,0,0,0,0,0,0]	0.0304 ± 0.0188	16.2855 ± 3.9058	0.5935 ± 0.1641	0.0728 ± 0.0414	12.0387 ± 2.4420
[1,1,1,1,1,1,1]	0.0090 ± 0.0065	$\mathbf{21.5895 \pm 3.4831}$	$\mathbf{0.7518 \pm 0.1211}$	$\mathbf{0.0390 \pm 0.0463}$	$\mathbf{15.9946 \pm 4.0459}$
[1,1,1,1,1,1,0]	0.0091 ± 0.0067	21.5496 ± 3.5169	0.7507 ± 0.1212	0.0394 ± 0.0463	15.9192 ± 4.0313
[1,1,1,1,1,0,0]	$\mathbf{0.0090 \pm 0.0063}$	21.5083 ± 3.3431	0.7499 ± 0.1209	0.0402 ± 0.0481	15.8599 ± 4.0589
[1,1,1,1,0,0,0]	0.0097 ± 0.0067	21.3205 ± 3.6386	0.7459 ± 0.1262	0.0433 ± 0.0386	14.9701 ± 3.6032
[1,1,1,0,0,0,0]	0.0161 ± 0.0088	18.8596 ± 3.2816	0.6846 ± 0.1396	0.1041 ± 0.0702	10.6228 ± 2.7887
[1,1,0,0,0,0,0]	0.0138 ± 0.0087	19.7032 ± 3.7333	0.6989 ± 0.1377	0.0661 ± 0.0276	12.1871 ± 1.9212
[1,0,0,0,0,0,0]	0.0162 ± 0.0113	19.0486 ± 3.8177	0.6870 ± 0.1343	0.0410 ± 0.0271	14.6198 ± 2.5092
[0,0,0,0,0,0,0]	0.0304 ± 0.0188	16.2855 ± 3.9058	0.5935 ± 0.1641	0.0728 ± 0.0414	12.0387 ± 2.4420

Figure A.5. Qualitative results from the Internal Connections Analysis on P2P($T_{2\rightarrow 2flair}$): channels are sequentially turned off. *Prediction* corresponds to config.[1111111]

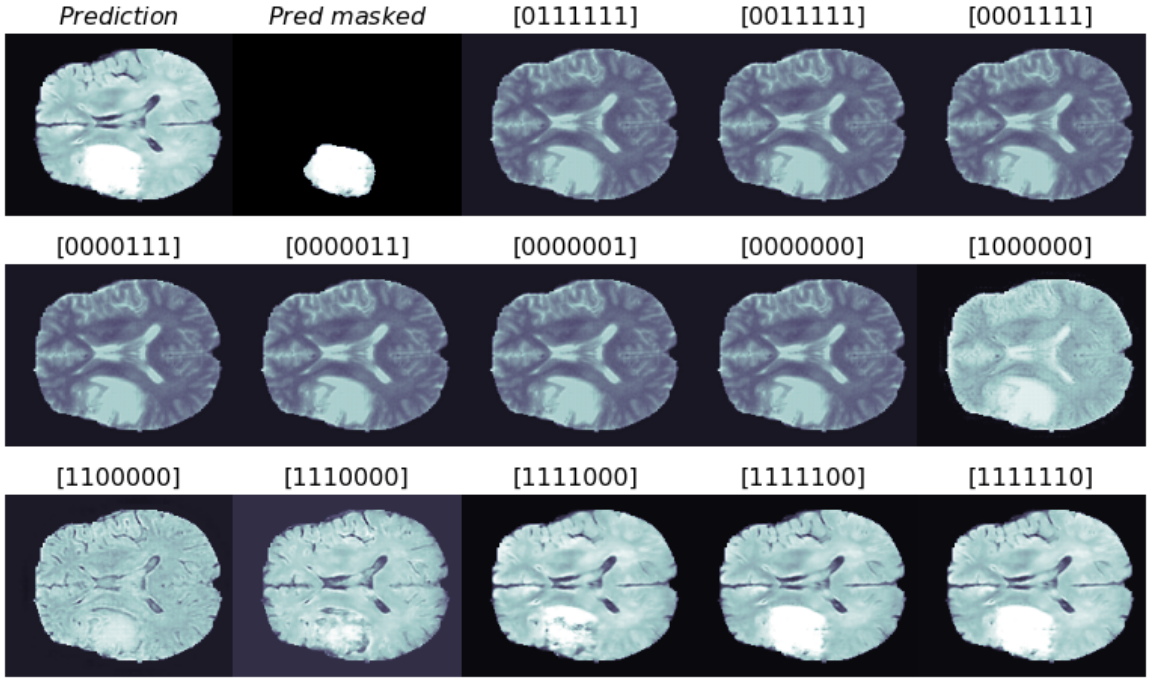


Table A.6. Quantitative results from the Internal Connections Analysis on MI-GAN$_{T_{2flair}}$: channels are sequentially turned off.

Internal off	**MSE**	**PSNR**	**SSIM**	**MSE$_{\text{tumor}}$**	**PSNR$_{\text{tumor}}$**
[1,1,1,1,1,1,1]	$\mathbf{0.0072 \pm 0.0050}$	$\mathbf{22.5524 \pm 3.5655}$	$\mathbf{0.7610 \pm 0.1175}$	$\mathbf{0.0258 \pm 0.0285}$	$\mathbf{17.4694 \pm 3.6137}$
[0,1,1,1,1,1,1]	0.0124 ± 0.0070	19.9894 ± 3.2877	0.6496 ± 0.1515	0.0391 ± 0.0177	14.5517 ± 2.1972
...	...	...	...	...	...
[0,0,0,0,0,0,0]	0.0124 ± 0.0070	19.9894 ± 3.2877	0.6496 ± 0.1515	0.0391 ± 0.0177	14.5517 ± 2.1972
[1,1,1,1,1,1,1]	$\mathbf{0.0072 \pm 0.0050}$	$\mathbf{22.5524 \pm 3.5655}$	$\mathbf{0.7610 \pm 0.1175}$	$\mathbf{0.0258 \pm 0.0285}$	$\mathbf{17.4694 \pm 3.6137}$
[1,1,1,1,1,1,0]	0.0073 ± 0.0050	22.5220 ± 3.5588	0.7604 ± 0.1171	0.0262 ± 0.0287	17.3608 ± 3.5861
[1,1,1,1,1,0,0]	0.0084 ± 0.0051	21.6656 ± 3.2445	0.7367 ± 0.1215	0.0287 ± 0.0311	16.8997 ± 3.4755
[1,1,1,1,0,0,0]	0.0100 ± 0.0065	21.0881 ± 3.6372	0.7374 ± 0.1233	0.0347 ± 0.0165	15.1478 ± 2.3996
[1,1,1,0,0,0,0]	0.0144 ± 0.0072	19.1951 ± 3.1252	0.6661 ± 0.1422	0.0536 ± 0.0470	13.8716 ± 3.3014
[1,1,0,0,0,0,0]	0.0107 ± 0.0072	20.8187 ± 3.6480	0.7339 ± 0.1241	0.0363 ± 0.0166	14.9462 ± 2.4227
[1,0,0,0,0,0,0]	0.0121 ± 0.0070	20.1199 ± 3.3328	0.6548 ± 0.1258	0.0526 ± 0.0197	13.1301 ± 1.8387
[0,0,0,0,0,0,0]	0.0124 ± 0.0070	19.9894 ± 3.2877	0.6496 ± 0.1515	0.0391 ± 0.0177	14.5517 ± 2.1972

Figure A.6. Qualitative results from the Internal Connections Analysis on MI-GAN$_{T_{2flair}}$: channels are sequentially turned off. *Prediction* corresponds to config.[1111111]