Universitat Autònoma de Barcelona

Dipòsit digital de documents de la UAB

This is the **published version** of the bachelor thesis:

Feliu Juarez, Oriol; Casas Roma, Jordi, dir. Generation of brain MRI images using Generative Adversarial Networks (GAN). 2023. (958 Enginyeria Informàtica)

This version is available at https://ddd.uab.cat/record/272819

under the terms of the BY-NC-ND license

Generation of brain MRI images using Generative Adversarial Networks (GAN)

Oriol Feliu Juárez

Abstract– This project aims to develop 3 different *Generative Adversarial Networks* (GAN) models with varying amount of layers (4, 7 and 15) that generate artificial images of brain MRI images, use different input values, such as amount of samples and epochs, and compare the results in terms FID score, time consumed, loss function and visual analysis. In this kind of machine learning model, two neural networks compete with each other in the form of a zero-sum game, with one network serving as the generator and the other as the discriminator. The generator is trained from a database that contains samples of brain MRI images, both healthy and sick ones, in NIfTI format, that are transformed into 2D images. Different neural networks and input parameters are tested and the results are compared. The project follows the CRISP-DM methodology, a widely used approach for machine learning. The results indicate that the 4-layered model is not complex enough to create realistic images when not using enough samples or epochs, while the 7-layered and 15-layered models are deep enough to generate good samples, the 7-layered one being the most balanced between computational resources and quality of generated images. Increasing the number and variety of samples is very important to create a model that can generate better samples, also rising the amount of epochs also helps to improve the results, but both of these practices increases by a great amount the time and resources consumed for training. The loss function also shows if the model is training correctly, the 4-layered model does not have a good progress of its loss value, as it does not get lower and the discriminators is not the opposite of the generator's, bot the 7-layered and 15-layered generator have a good evolution of the loss value. The loss evolution improves by using bigger datasets for training.

Keywords– Generative Adversarial Networks, GAN, artificial intelligence, tensorflow, keras, MRI, machine learning, image generation, deep learning, synthetic data, hyperparameter tuning

Resum– Aquest projecte té com a objectiu desenvolupar 3 models diferents de Xarxes Adversàries Generatives (GAN) amb una quantitat variable de capes (4, 7 i 15) que generen imatges artificiales d'imatges de ressonàncies magnètiques cerebrals, utilitzant diferents valors d'entrada, com la quantitat de mostres i èpoques, i comparar els resultats en termes de puntuació FID, temps consumit, funció de pèrdua i anàlisi visual. En aquest tipus de model d'aprenentatge automàtic, dues xarxes neurals competeixen entre elles en forma d'un joc de suma zero, amb una xarxa actuant com a generador i l'altra com a discriminador. El generador s'entrena a partir d'una base de dades que conté mostres d'imatges de ressonàncies magnètiques cerebrals, tant saludables com malaltes, en format NIfTI, que es transformen en imatges 2D. S'avaluen diferents xarxes neurals i paràmetres d'entrada i es comparan els resultats. El projecte segueix la metodologia CRISP-DM, un enfocament ampliament utilitzat per a l'aprenentatge automàtic. Els resultats indiquen que el model de 4 capes no és prou complex per crear imatges realistes quan no s'utilitzen prou mostres o èpoques, mentre que els models de 7 i 15 capes són prou profunds per generar bones mostres, sent el de 7 capes el més equilibrat entre els recursos computacionals i la qualitat de les imatges generades. Augmentar el nombre i la varietat de mostres és molt important per crear un model que pugui generar millors mostres, també l'augment de la quantitat d'èpoques ajuda a millorar els resultats, però ambdues pràctiques augmenten en gran mesura el temps i els recursos consumits per a l'entrenament. La funció de pèrdua també mostra si el model està entrenant correctament, el model de 4 capes no té un bon progrés del seu valor de pèrdua, ja que no es redueix i el discriminador no és l'oposat del generador, però el generador de 7 i 15 capes té una bona evolució del valor de pèrdua. L'evolució de la pèrdua millora en utilitzar conjunts de dades més grans per a l'entrenament.

Paraules clau– Xarxes Adversàries Generatives, GAN, intel·ligència artificial, TensorFlow, Keras, ressonància magnètica, aprenentatge automàtic, generació d'imatges, aprenentatge profund, dades sintètiques, afinació de hiperparàmetres. (GAN)

1 INTRODUCTION

THIS document is the final degree project report, which aims to generate artificial images using *Generative Adversarial Networks* (GAN). GANs are a type of machine learning algorithms designed by Ian Goodfellow and his team in 2014, therefore, it is a recent technology that is still being researched. In this model, two neural networks compete in the form of a zero-sum game, meaning the loss of one is the gain of the other and vice versa. In this model, two neural networks, one that creates fake data, called the *generator*, and another that has to detect whether the data it receives is real or generated, called the *discriminator*.

The goal of this project is to create three different GAN models that generate brain MRI images. The generator models have different architectures with varying numbers of layers to investigate the impact on the quality of generated images. The models are trained using the same database that contains different images of healthy and ill brains. The results are summarized in tables that compare the model used, time consumed, number of samples and epochs used for training and Fréchet Inception Distance (FID) score. This thesis seeks to contribute to the understanding of how the architecture of GAN generator models affects their performance in generating 2D brain MRI images.

Brain MRI images are performed using a scanner and can detect different conditions such as, among others, internal hemorrhages, inflammation, problems with brain development, tumors, infections or damage caused by injuries or embolism. Therefore, they are a tool that medical staff can use to find the cause of a problem related to the brain.

This project uses a database with NIFTI format samples, which hold 2D images of brain slices. The aim is to extract relevant samples from these files to train the Generative Adversarial Network (GAN). The training dataset contains real images like the sample displayed in figure 1.

There is a lack of data, especially of healthy brains, because it is an invasive and unpleasant process for the patient. GAN can help create more samples that can be used to train machine learning models that do not have enough data.

The code used in this project can be found at the author's Git Hub [1]. It is a Jupyter notebook, uses Python and can be used with any dataset in NIFTI or JPG format. All trained GAN models, with different layers, amount of samples and epochs can be found at the author's Google Drive [2].

2 STATE OF THE ART

The aim of this section is to show the most recent research and projects related to GANs and their use in the medical field, specifically in the generation of MRI images.

In 2014, computer scientist Ian J. Goodfellow and his colleagues published the first GAN machine learning algorithm and the first article on this topic (DOI reference 10.1186) [3]. This marked the first iteration of two neural networks competing with each other to generate artificial images.

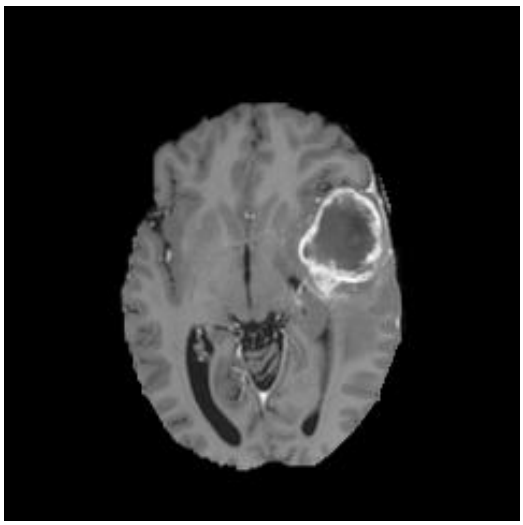

Fig. 1: Example of real brain MRI of a brain with tumor used for model training.

Since the publication of this first version of GAN, multiple different versions have been created that implement different types of neural networks, such as CGAN (Conditional Generative Adversarial Network), DCGAN (Deep Convolutional Generative Adversarial Network) or ProGAN (Progressive growing of Generative Adversarial Networks).

There are multiple libraries that can be used to model GANs, two of the most important are keras [4] and pyTorch [5]. These two libraries are among the most famous for working with machine learning and have also developed GAN models that allow for easy implementation.

GANs have been successful in various tasks, including generating images, sounds and synthetic text. They have also been used to improve image resolution, translations between different languages and synthetic speech.

This technology has been applied in various areas of medicine, with the goal of generating synthetic medical data, improving the accuracy of diagnoses, and increasing the data available for training machine learning models that have insufficient data. For example, GANs have been used to generate mammograms, CT scans, and MRI images of different parts of the human body.

Different applications of GANs in brain magnetic resonances can be found. In 2022, Hazrat Ali and his colleagues published an article (with DOI reference 10.1186) in which GANs are used with brain MRI images [6], they explain how this machine learning algorithm can be used to improve the already used AI in brain MRI images, since it allows data augmentation of MRI datasets and help with the segmentation of tumors in brains.

Researcher Rajeev K. Gupta and his colleagues explain in their article (with DOI reference 10.1007) [7] how the application of GANs to increase the number of samples in a brain MRI database can significantly increase the accuracy of a predictive model, specifically it contributes to achieve 98 % of accuracy. In this article it is specified that a C-GAN (Conditional Generative Adversarial Networks) is used.

- Contact e-mail: oriol.feliu9@gmail.com
- Specialization: Computing
- Work tutored by: Jordi Casas Roma (Computing)
- Course 2022/23

In this project, the theory and code provided in the book "Introducción Práctica con Keras y Tensorflow 2" by Jordi Torres [8]. This book provided a comprehensive explanation of the concepts and practical implementation of deep neural networks and GAN using the Keras and Tensorflow libraries. The same code used and modified for the creation of the different models evaluated in this work can be found in the author's Git Hub repository [9]. The GAN that J. Torres creates in his work uses the MNIST dataset, which is a large database widely used for training and testing image classification and generation models; the images are 28x28 pixels and only have one channel, so with a few changes the generator and discriminator are modified to produce and judge 112x112 grayscale images.

Deep learning models are the core of the GAN architecture, as it can use deep feedforward networks, recurrent neural networks, convolutional neural networks, deep belief networks, and deep reinforcement learning. The online book "Deep Learning Book" [10], aimed at students, researchers and developers, provides a comprehensive introduction, mathematical foundations, implementation details and practical considerations for building deep learning models.

GANs have proven to be a very effective tool for generating artificial data and have great potential in fields such as computer vision, natural language processing and speech synthesis. This neural network algorithm has great potential in the medical field, especially for doing artificial data augmentation of medical samples. However, it is important to thoroughly evaluate the performance of GANs on medical data and ensure that they are reliable before using them in clinical settings.

3 Objective

The main objective of this project is to create three different GAN models capable of generating synthetic images of brain MRI that are trained used real images. This objective can be broken down into the following secondary objectives:

- Create a convolutional network that can differentiate between a real MRI image and an artificially generated one, this network is called *discriminator*.
- Create three convolutional networks with different architectures that can generate artificial brain MRI images, called *generators*.
- Force the generator and discriminator to compete against each other in a zero-sum game, that is, what one loses is the gain of the other and vice versa. This competition will be repeated until the generator network can create artificial MRI images that seem real.
- Code different evaluating metrics such as loss of the model and FID score that compares real and generated images.
- Test various GAN models with differing architectures, including a varying number of layers and different input values such as the number of epochs used during training. The results of these models will be compared and evaluated using comparison tables and confusion matrices to determine which model is the most effective at generating images. Also make visual analysis of the generated samples.
- Learn to work with 3D brain MRI images, in this case in *NIFTI* format and to process them to obtain a set of 2D images, correctly aligned, segmented and resized.

4 Planning

This project uses the *Cross Industry Standard Process for Data Mining* (CRISP-DM) [11] methodology, as it is one of the most suitable methodologies for projects related to machine learning and data mining. It is a process model that provides a structured approach to planning, implementing and deploying machine learning projects. It is designed to be flexible and adaptable to different domains and project types.

The following phases are followed:

1. **Data Acquisition and Storage**: The objective of this phase is to acquire the initial data and store it in a database that can be used in the project. This phase is important as it sets the foundation for the rest of the project by ensuring that the data is properly collected, stored, and governed.

2. **Descriptive Analysis and Data Preprocessing**: This phase is focused on understanding the characteristics of the data and preparing it for modeling. Data cleaning is performed, ensuring that the data is valid by modifying or discarding it when appropriate. Finally, the data is transformed into a format that is more suitable for modeling, this includes tasks such as normalization, standardization, and encoding categorical variables.

3. **Modeling**: This phase aims to define and create the neural networks used for the machine learning algorithm and analyze the data using the created models.

4. **Optimization**: Different types of neural networks and input values are used to obtain the best results and determine the best combination.

5. **Evaluation of results**: In the final step of the CRISP-DM methodology, the goal is to find the most suitable models and parameters for this project.

6. **Report Writing**: This phase is the final step of the project and it's focused on documenting and communicating the work that has been done and the results obtained during the project. Once this report is finished, a presentation of this project will be made and delivered to a court.

   Figure 2 displays the project schedule represented in a Gantt chart made with Microsoft Project. The project began on October 10th, 2022 and ended on January 21st, 2023, lasting a total of 81 days.

	Task Mode	Task Name	Duration	Start	Finish
1		**Generation of brain MRI images using GAN**	**81 days**	**Mon 10/10/22**	**Tue 31/01/23**
2		**Data Acquisition and Storage**	**6 days**	**Mon 10/10/22**	**Mon 17/10/22**
3		Find MRI images datsaet	2 days	Mon 10/10/22	Tue 11/10/22
4		Download data	2 days	Wed 12/10/22	Thu 13/10/22
5		Upload data to Google Drive	2 days	Fri 14/10/22	Mon 17/10/22
6		**Descriptive Analysis and Data Preprocessing**	**15 days**	**Fri 14/10/22**	**Thu 03/11/22**
7		Understand nifti format and how to transform to jpg	2 days	Fri 14/10/22	Mon 17/10/22
8		Code to transfrom nifti to jpg	5 days	Tue 18/10/22	Mon 24/10/22
9		Nifti to jpg and store to Google Drive	2 days	Tue 25/10/22	Wed 26/10/22
10		Data cleaning	2 days	Thu 27/10/22	Fri 28/10/22
11		Data transformation and normalization	4 days	Mon 31/10/22	Thu 03/11/22
12		**Modeling**	**21 days**	**Fri 04/11/22**	**Fri 02/12/22**
13		Design generator	7 days	Fri 04/11/22	Mon 14/11/22
14		Design discriminator	7 days	Tue 15/11/22	Wed 23/11/22
15		Model training	7 days	Thu 24/11/22	Fri 02/12/22
16		**Optimization**	**21 days**	**Mon 05/12/22**	**Mon 02/01/23**
17		Run different types of generator	7 days	Mon 05/12/22	Tue 13/12/22
18		Run different types of discriminator	7 days	Wed 14/12/22	Thu 22/12/22
19		Run different input parameters	7 days	Fri 23/12/22	Mon 02/01/23
20		**Evaluation**	**13 days**	**Tue 03/01/23**	**Fri 20/01/23**
21		Code evaluation methods	4 days	Tue 03/01/23	Mon 09/01/23
22		Code FID score	2 days	Tue 03/01/23	Wed 04/01/23
23		Run missing models and parameters	5 days	Tue 10/01/23	Mon 16/01/23
24		Make tables with all the results	4 days	Tue 17/01/23	Fri 20/01/23
25		**Report Writing**	**7 days**	**Mon 23/01/23**	**Tue 31/01/23**
26		Create images and tables for the report	4 days	Mon 23/01/23	Thu 26/01/23
27		Final report	7 days	Mon 23/01/23	Tue 31/01/23

Fig. 2: Gantt chart of the project planning.

5 Methodology and Results

As described in the Planning section, this project follows the CRISP-DM methodology. In this segment, it is explained in depth what has been done in each phase.

5.1 Data Acquisition and Storage

The data used in this project is sourced from the *Decathlon* database [12]. The database contains MRI images of various body parts including the brain, heart, lungs and colon, and is created to support the development of various types of machine learning models. After downloading the data, it is stored in Google Drive and can be accessed via a Google Colab notebook.

5.2 Descriptive Analysis and Data Preprocessing

The database of brain MRI images consists of samples in the NIFTI (Neuroimaging Informatics Technology Initiative) format, commonly used in medical imaging to store the results of MRI or CT scans. The database contains 150 slices of the human brain, but only slices 60 to 100 are considered relevant for training the generator model, as they provide the best view of the brain. The samples include both healthy brains and brains with one or multiple tumors.

The library med2image [13] is used to convert the NIFTI samples to JPG. The initial dataset consists of 912 NIFTI samples, each with 4 frames and 150 slices. After converting to JPG, the dataset contains 145,929 brain MRI images.

The images have originally a size of 240x240, in order to make the execution of GANs more efficient and faster, a dimension change to 112x112 is applied. These are grayscale images, so the images can be represented with a two-dimensional array and, in addition, these are normalized in order to transform from a range of 0 to 255 to a range of -1 to 1. The values of the array are floating type.

5.3 Modeling

The GAN is composed of two main components: a generator and a discriminator. The generator is a neural network that is trained to generate new data samples that are similar to the training set. It takes a random noise vector as input and generates a new data sample as output. The discriminator is another neural network that is trained to distinguish between real and generated data samples. It takes a data sample as input and outputs a probability that the sample is real. To understand this concept better, figure 3 is a graphical representation of this algorithm that uses real and generated samples of this project:

The generators and the discriminator are coded using the TensorFlow Keras library.

Three different generator models have been tested, the number of layers used differs in each one. The models are: 4 layered, 7 layered and 15 layered. All of them have in common the following:

- They have sequential structure.
- The first layer is a Dense layer that takes a random input and a one-dimensional array of 100 elements, which is usually called latent space.
- The output is a convolutional 2D transpose layer that generates a 112x112 image of a single channel (grayscale), that uses a tanh activation function to ensure that the generated image pixels are in the range [-1, 1].

The simplest generator has 4 layers. Besides the Dense input layer and the convolutional 2D transpose output layer, it only has a leaky ReLU activation function and a reshape layer. This neural network has 20,072,001 trainable parameters. The model is summarized in table 1.

TABLE 1: 4 LAYERED GENERATOR

Layer (type)	Output Shape	Param #
Dense	(None, 200704)	20070400
LeakyReLU	(None, 200704)	0
Reshape	(None, 56, 56, 64)	0
Conv2DTranspose	(None, 112, 112, 1)	1601

The second generator has 7 layers in total. It processes the input through a series of dense, reshape and transposed convolutional layers with batch normalization and leaky

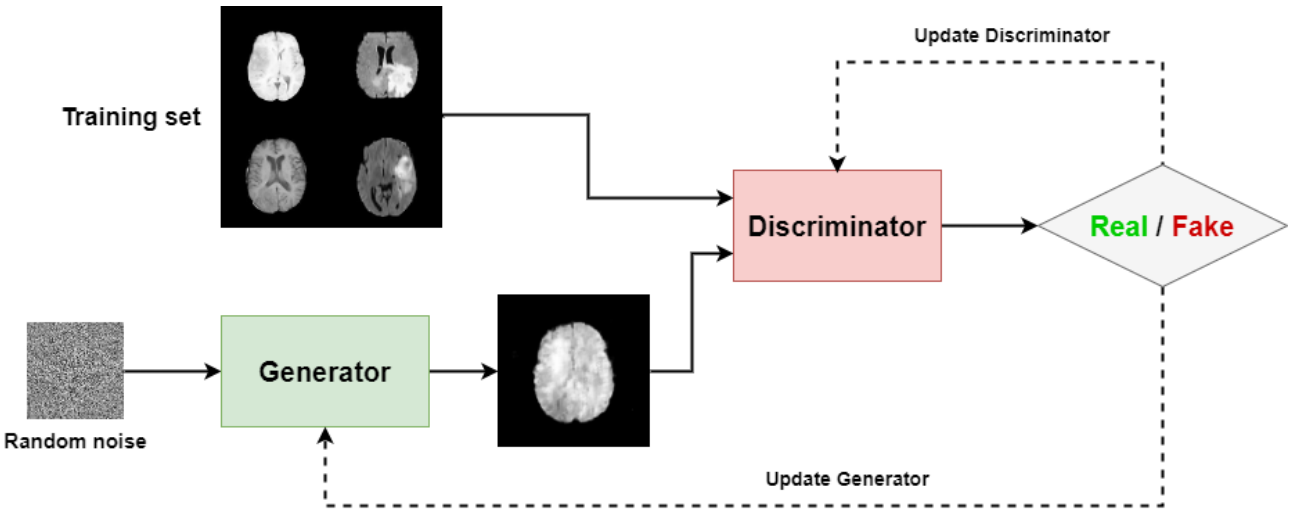

Fig. 3: Graphical representation of the GAN algorithm with real and generated images of this project.

ReLU activation functions. It has a total of 21,096,577 trainable parameters. The model is summarized in table 2.

TABLE 2: 7 LAYERED GENERATOR

Layer (type)	Output Shape	Param #
Dense	(None, 200704)	20070400
Reshape	(None, 28, 28, 256)	0
Conv2DTranspose	(None, 56, 56, 128)	819328
BatchNormalization	(None, 56, 56, 128)	512
LeakyReLU	(None, 56, 56, 128)	0
Conv2DTranspose	(None, 56, 56, 64)	204864
BatchNormalization	(None, 56, 56, 64)	256
LeakyReLU	(None, 56, 56, 64)	0
Conv2DTranspose	(None, 112, 112, 1)	1601

The deepest generator has 15 layers. It has a reshape layer and multiple convolutional, batch normaliztion and leaky ReLu layers. This neural network has a total of 8,092,929 trainable parameters. The model is summarized in table 3.

TABLE 3: 15 LAYERED GENERATOR

Layer (type)	Output Shape	Param #
Dense	(None, 50176)	5017600
Reshape	(None, 14, 14, 256)	0
Conv2DTranspose	(None, 28, 28, 256)	1638656
BatchNormalization	(None, 28, 28, 256)	1024
LeakyReLU	(None, 28, 28, 256)	0
Conv2DTranspose	(None, 28, 28, 128)	819328
BatchNormalization	(None, 28, 28, 128)	512
LeakyReLU	(None, 28, 28, 128)	0
Conv2DTranspose	(None, 56, 56, 128)	409728
BatchNormalization	(None, 56, 56, 128)	512
LeakyReLU	(None, 56, 56, 128)	0
Conv2DTranspose	(None, 56, 56, 64)	204864
BatchNormalization	(None, 56, 56, 64)	256
LeakyReLU	(None, 56, 56, 64)	0
Conv2DTranspose	(None, 112, 112, 1)	1601

Only one kind of model is used for the discriminator, it is a sequential neurnal network with 10 layers. The input of the model is a 1 channel 112x112 image, which is passed through several convolutional layers. The activation function used in these convolutional layers is LeakyReLU with an alpha value of 0.01, this activation function helps to avoid the vanishing gradient problem. The model also has batch normalization layers and the final layer is a dense layer with a single output node and a sigmoid activation function. The output of this layer represents the probability that the input image is real or fake. It has 282,497 trainable parameters. The model is summarized in table 4.

TABLE 4: 10 LAYERED DISCRIMINATOR

Layer (type)	Output Shape	Param #
Conv2D	(None, 56, 56, 32)	832
LeakyReLU	(None, 56, 56, 32)	0
Conv2D	(None, 28, 28, 64)	51264
BatchNormalization	(None, 28, 28, 64)	1601
LeakyReLU	(None, 28, 28, 64)	1601
Conv2D	(None, 14, 14, 128)	1601
BatchNormalization	(None, 14, 14, 128)	1601
LeakyReLU	(None, 14, 14, 128)	1601
Flatten	(None, 25088)	1601
Dense	(None, 1)	1601

5.4 Optimization

The batch size can affect the performance of both models. For the generator, a larger batch size can result in better results as it can make use of more information from the data distribution, but it can also require more computational resources because it requires more memory and a larger model. For the discriminator, a larger batch size can improve results as it can obtain more diverse samples to learn from, but it can face the same problems as the generator.

The learning rate determines the speed at which the model learns during training, a higher rate means the model will learn faster, but it may make the model more prone to overfitting, whereas a lower rate translates into slower learning, but it may result in a more stable model and less

prone to overfitting. The performance of the learning rate is evaluated with the loss of the models. Finding the optimal rate can be a trial and error process.

An Adam optimizer is used for the neural networks, which is a gradient-based optimization algorithm widely used in machine learning, this optimizer is an extension of the stochastic gradient descent optimization algorithm (SGD). In this project, only the learning rate is introduced as a parameter of the optimizer and it is considered that the default values of the rest of the parameters do not need to be modified.

As explained in the modeling section, 3 different model architectures are evaluated, each one with more layers than the other. The number of layers in the generator of a GAN can have a significant impact on its performance and results. A deeper generator has the ability to learn and reproduce more intricate features, such as brain tumors, resulting in higher-quality generated samples. However, this advantage comes at a cost as it requires a larger amount of training data and computing resources, and there is a risk of overfitting. On the other hand, a generator with fewer layers may struggle to accurately capture the intricacies of the training data and produce low-quality generated samples.

Different values of epochs and number of samples for the training phase are evaluated. A larger number of epochs and samples probably leads to better performance and generated images of a higher quality. Too many epochs and not enough samples can lead to overfitting, but training for too few epochs can result in underfitting.

Determining the most appropriate number of layers and epochs for a GAN requires taking into account various trade-offs while also considering factors such as the size and complexity of the training data, the computational resources that are available and the desired results.

5.5 Evaluation of results

In this phase, the results of the different model architectures and number of epochs are put in tables for its evaluation and compared to find the best combination for image generation and checked to meet the main objectives. Loss functions and FID score are used to find an objective value of the GAN's performance. The generated images will also be displayed. Furthermore, errors are identified and corrected.

5.5.1 Loss function

The loss function is used to train both the generator and the discriminator. The two models have opposing loss values, which confirms what was previously explained about how GAN works: the two models compete with each other in a zero-sum game.

The differences between the evolution of the loss values of each generator model and the amount of samples used for training are very notable. Figure 4 displays the loss evolution of the 15-layered generator and 50 thousand samples for training, it is an optimal progress of the loss function because it starts with a very high value and with each epoch gets lower and less steeper. Figure 5 is the evolution of the 7-layered generator with 30 thousand samples for training, it is a less optimal evolution because in its first epochs the loss value fluctuates a lot and the curve of the progress it's not notable enough. Figure 6, displaying the line graph of the 4-layered generator and only 10 thousand samples, shows how a model that does not have enough depth and samples for training may end up not working properly, in this case the discriminator's loss is not the negative of the generator's, the evolution of the loss functions is incorrect, as it does not get lower with each epoch, this may mean that the model is having overfitting.

In this case, the deepest generator, with 15 layers, and the 50 thousand samples for training show the best results for the loss function, in other words, it is the model that is learning more correctly. In addition to this, all models display a somewhat correct progress when using enough samples.

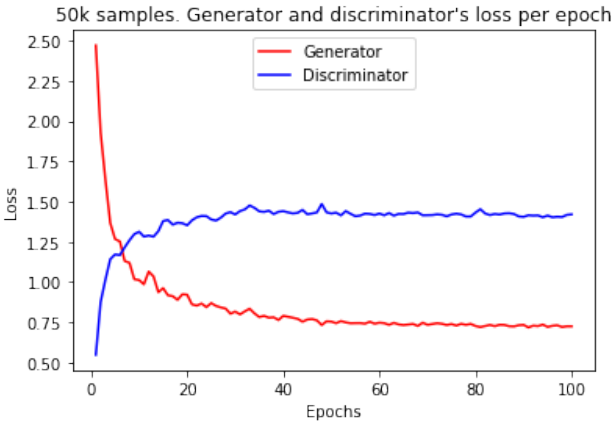

Fig. 4: Line graph showing the evolution of the discriminator and the 15-layered generator loss over 100 epochs using 50k samples for training.

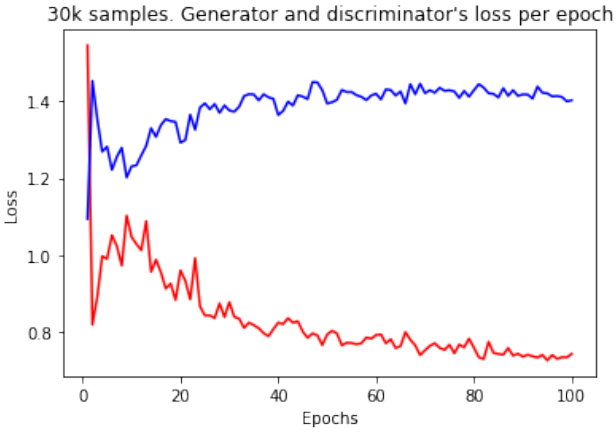

Fig. 5: Line graph showing the evolution of the discriminator and the 7-layered generator loss over 100 epochs using 30k samples for training.

5.5.2 FID score and time consumed

Fréchet Inception Distance (FID) score is a metric commonly used in the evaluation of the samples generated by a GAN. It measures the similarity between two images, in this case, the real and the generated images. The closer this value is to 0, the more similar are the generated images to the real ones. A high value means that both sets are significantly different from each other. FID score is a useful

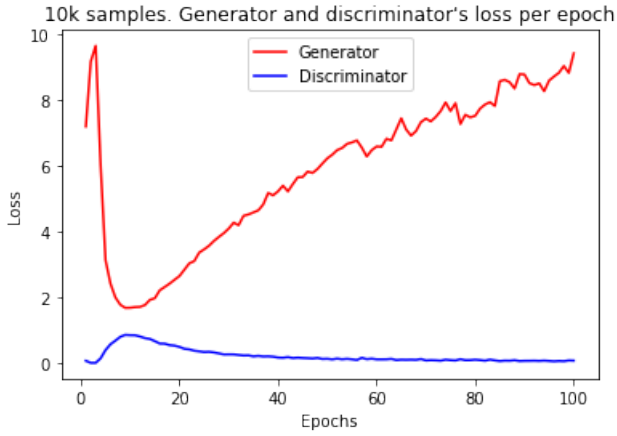

Fig. 6: Line graph showing the evolution of the discriminator and the 4-layered generator loss over 100 epochs using 10k samples for training.

metric for GAN evaluation because it considers both quality and diversity of the generated images. For this project, a modified version of the code created by Jason Brownlee found in a *Machine Learning Mastery* article [14] is used for calculating FID score.

Table 5 provides a comparison of the results obtained from the three different generator models with varying number of layers (4, 7, and 15), three different size of samples (10k, 30k and 50k) and three different amounts of epochs (20, 60 and 100). The time consumed during the training process is used as a metric to evaluate the resources consumed, while the FID score was used to measure the quality of the generated images. In addition to the quantitative results, images generated by each model are presented to provide a visual evaluation of the results.

The table provides the following information:

- The generator model with 4 layers performed worse than the other architectures, it has the higher FID in all cases.
- The 7-layered generator gave better results than the 4-layered model, but worse than the deepest, the model with 15 layers.
- As expected, the model with 15 layers generated the best images out of the three generators.
- More layers and more samples for training almost always traduces to better performance, lower FID score, and a much higher cost of computational resources, for example, training with 10 thousand samples takes approximately 1 hour and training with 50 thousand takes about 4 hours.

5.5.3 Visual analysis

In this subsection, a visual evaluation of the samples produced by the generators is made, this type of analysis does not provide quantitative data on the accuracy of the model, but it allows the subjective judgement of whether the images generated by the model appear to be authentic.

TABLE 5: RESULTS

GAN	Samples	Epochs	Time	FID
4 layers	10k	50	12 m	42.84
4 layers	30k	50	29 m	41.47
4 layers	50k	50	57 m	39.98
7 layers	10k	50	20 m	43.45
7 layers	30k	50	42 m	39.72
7 layers	50k	50	1 h 52 m	41.92
15 layers	10k	50	37 m	43.2
15 layers	30k	50	1 h 47 m	42.12
15 layers	50k	50	2 h 12 m	41.34
4 layers	10k	100	23 m	42.81
4 layers	30k	100	43 m	41.9
4 layers	50k	100	60 m	39.83
7 layers	10k	100	37 m	42.84
7 layers	30k	100	1 h 17 m	38.47
7 layers	50k	100	3 h 48 m	39.83
15 layers	10k	100	59 m	42.9
15 layers	30k	100	2 h 32 m	43.28
15 layers	50k	100	4 h	39.16

Figure 7 shows a sample generated by the model with 4 layers, using 10 thousand samples and 20 epochs for training. These values are the lowest tested in this project and it proves that not having enough epochs and samples for training leads to a bad performance of the GAN model. This sample shows clear errors like grey and white outside of the brain, the brain itself does not have a correct shape and its different sections are not differentiable. This sample makes sense with the FID score shown in the results table, that has the worst score out of all the model and input values combinations.

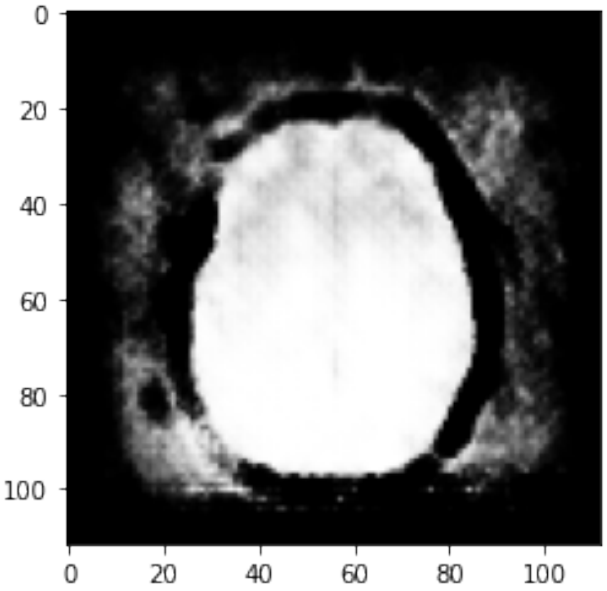

Fig. 7: Sample generated by the 4-layered generator, using 10 thousand samples and 20 epochs for training.

Training the model with 4 layers for 50 or 100 epochs and 30 or 50 thousand samples provide much better results. Figure 8 shows a good sample generated by this model, displaying much better results with a clearer shape and differ-

entiated regions of the brain, free of noise. It must be noted that many generated samples have errors such as wrong shape and noise, so this model is not stable, producing results of varying quality.

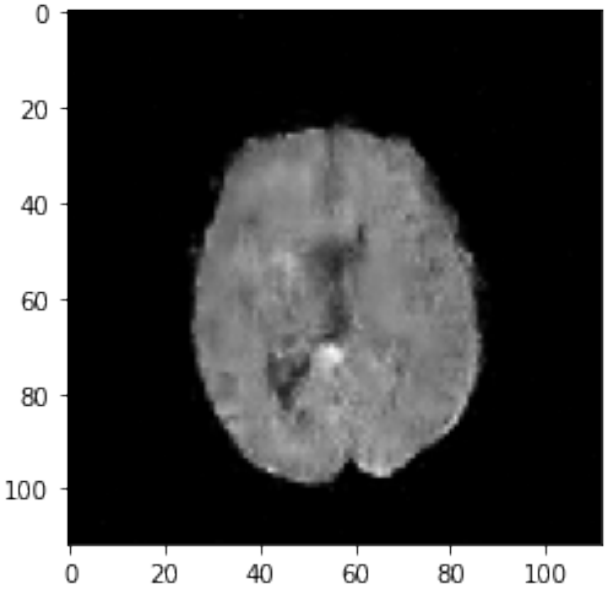

Fig. 8: Sample generated by the 4-layered generator, using 50 thousand samples and 100 epochs for training.

Figure 9 displays an image generated by the 7-layered model, using 50 thousand samples and 100 epochs for training. This model, provides a much better result than the 4-layered model, it has a clearer brain shape, no noise around it and different areas of the brain noticeable, even though these are not correctly separated. Some of the samples created by this model are not realistic.

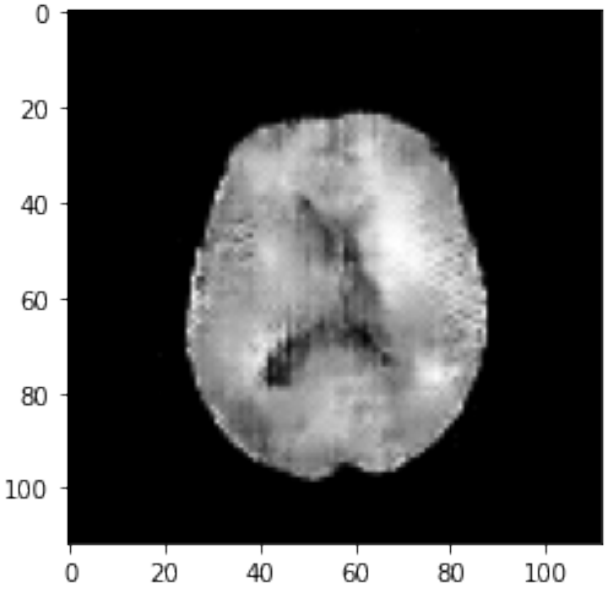

Fig. 9: Sample generated by the 7-layered generator, using 50 thousand samples and 100 epochs for training.

Figure 10 displays a image generated by the 15-layered model, using 50 thousand samples and 100 epochs for training. These values are the best tested in this project and it shows the best results in the visual analysis. This image has a more similar shape to a real brain, there is no noise around it, different parts of the brain are noticeable and it evens shows a small black dot in its left side that resembles a tumor, similar to the ones found in the real dataset.

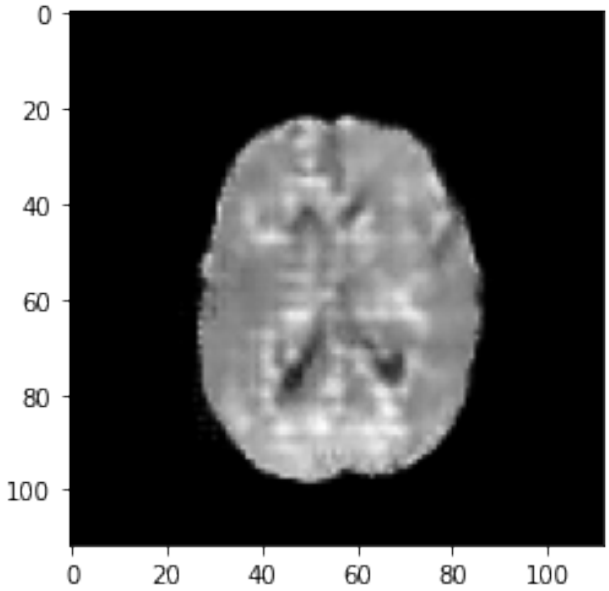

Fig. 10: Sample generated by the 15-layered generator, using 50 thousand samples and 100 epochs for training.

Using a more complex model, as expected, provides better results, a lower FID score than the others and a generates an image that is more similar to the real samples. The one aspect that proved to have a bigger impact in the results is the number of samples used for training, as in each model performed better with more samples. Also rising the amount of epochs generally gave better results.

One of the problems faced in this project is not having enough amount and variety of data, as it is necessary to create a GAN model able to generate images indistinguishable from reality. This would have been impossible even having the optimal dataset for training, the computational resources available were not powerful enough, this can be seen in the results table, as using 50 thousand samples required multiple hours of training.

6 Conclusions

The objectives set at the beginning of the project were met, all GAN models were developed and performed as intended, they were able to generate somewhat realistic brain MRI images.

The loss value line graphs obtained made sense with the generated images and results of the evaluations. The 4-layered one couldn't have correct a progress of the loss when training with not enough samples, as it does not have a deep enough architecture. the loss graphs of the other two models showed expected progression, always getting lower and the discriminator's loss being the opposite of the generator's. Without enough samples the loss function proved to not be stable and in some cases fluctuate, specially with smaller training datasets.

The model with 4 layers proved to be the worst, it got the worst results of FID score and generated images that weren't very realistic. With enough samples and epochs, it could generate samples with the correct shape and color, but it could not differentiate between different regions of the brain or avoid creating noise around it. Also, this model is very unstable, producing samples of varying quality, with some being realistic and others not even having a similar shape or free of noise.

The 7 layered and 15-layered generator models gave the best results, using the highest amount of samples and epochs, in this case 50 thousand and 100, provided the best training for the model and, consequently, better results, such as better FID score and generated images of better quality. Both proved to be good architectures, but the 7-layered model consumed a lot less computational resources and time, so it gives a good balance between computational resources and quality of generated images.

The number of samples in the training dataset was found to be a crucial factor in creating a model that can generate realistic samples. Using 30 or 50 thousand real images for training improved FID score by a notable amount, but the results were more obvious when doing a visual analysis. Increasing the amount of samples came at the cost of much greater computational resources and time, for example, training the 15-layered generator with 10 thousand samples and 100 epochs took about 1 hour, but using 50 thousand images took 4 hours.

FID score is a useful quantitative measure for evaluating GANs, but additional methods, such as Inception Score and Precision-Recall curve, may be necessary to fully evaluate the models. Also, FID score is not capable of knowing if the shape of the brain is correct, the areas inside it are well places and differentiated or if it displays any conditions like tumors, in this case, a visual analysis is better because it can do all the things mentioned before, even though it is not capable of providing an objective quantitative value.

Future thesis projects can expand on the research of creating a GAN that generates brain MRI images by exploring other related topics such as:

- Enhancing the quality of generated MRI images.
- GAN that generates images of other body parts, such as knee MRI image.
- Use the GAN in machine learning models that uses real images and see if the incorporation of generated images improves aspects of the model such as accuracy.
- A GAN that generates images in other formats, such as 3D images.
- Evaluation of GAN for real clinical applications.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my mentor, Jordi Casas Roma, who guided me to be able to complete this project. I would also like to thank my friends and family who have supported me and provided their insights on this work.

REFERENCES

[1] O. Feliu, "OriolFeliu/GAN-BRAIN-MRI," GitHub, 7-Feb-2022. [Online]. Available: https://github.com/OriolFeliu/GAN-BRAIN-MRI. [Accessed: 7-Feb-2022].

[2] O. Feliu, "models," Google Drive, 7-Feb-2022. [Online]. Available: https://drive.google.com/drive/folders/1F6xIUdmXOS24hYpcMaKNCUheblKS0CjR?usp=share_link. [Accessed: 7-Feb-2022].

[3] I. Goodfellow et al. "Generative adversarial nets", 2014. [Online] Available at: https://arxiv.org/abs/1406.2661

[4] E. Linder-Norén, "Eriklindernoren/Keras-Gan: Keras implementations of generative adversarial networks," GitHub, 06-Jan-2021. [Online]. Available: https://github.com/eriklindernoren/Keras-GAN. [Accessed: 11-Nov-2022].

[5] GitHub - eriklindernoren/PyTorch-GAN: PyTorch implementations of Generative Adversarial Networks. (n.d.). GitHub. Retrieved October 9, 2022, from https://github.com/eriklindernoren/PyTorch-GAN

[6] H. Ali et al. et al. "The role of generative adversarial networks in brain MRI: a scoping review", 2022. [Online] Available: https://doi.org/10.1186/s13244-022-01237-0

[7] R. K. Gupta et al. "Brain Tumor Detection and Classification Using Cycle Generative Adversarial Networks", 2022. [Online] Available: https://doi.org/10.1007/s12539-022-00502-6

[8] J. Torres, *Python deep learning: introducción práctica con Keras y TensorFlow 2*, 1st ed. Barcelona, Spain: Marcombo, 2020.

[9] J. Torres, "JORDITORRESBCN/Python-Deep-Learning: Código del Libro Python Deep Learning," GitHub, 07-Jan-2019. [Online]. Available: https://github.com/jorditorresBCN/python-deep-learning. [Accessed: 10-Nov-2022].

[10] I. Goodfellow, Y. Bengio & A. Courville. *Deep Learning*. USA: MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/

[11] C. Schröerab, F. Kruseb & J. Marx "A Systematic Literature Review on Applying CRISP-DM Process Model", Procedia Computer Science, 181, 526–534, 2021. [Online] Available: https://doi.org/10.1016/j.procs.2021.01.199

[12] A. L. Simpson et al. "Medical Segmentation Decathlon - Generalisable 3D Semantic Segmentation" February 25, 2019 [Dataset]. Available: https://arxiv.org/abs/1902.09063

[13] R. Pienaar, "FNNDSC/med2image: Converts Medical Images to more displayable formats, e.g. NIfTI to JPG.," GitHub, 28-Apr-2022. [Online]. Available: https://github.com/FNNDSC/med2image. [Accessed: 14-Nov-2022].

[14] J. Brownlee, "How to implement the Frechet Inception Distance (FID) for evaluating Gans," MachineLearningMastery.com, 10-Oct-2019. [Online]. Available: https://machinelearningmastery.com/how-to-implement-the-frechet-inception-distance-fid-from-scratch/. [Accessed: 02-Dec-2022].