Universitat Oberta de Catalunya (UOC)

Máster Universitario en Ciencia de Datos (*Data Science*)

TRABAJO FINAL DE MÁSTER

Area: Medicine

Autor: Miguel Tablado

Tutor: Baris Kanber

Profesor: Ferran Prados Carrasco

Barcelona, January 15, 2023

FICHA DEL TRABAJO FINAL

Título del trabajo:	Artificial MRI brain images creation with Variational Autoencoders
Nombre del autor:	Miguel Tablado León
Nombre del Tutor/a de TF:	Baris Kanber
Nombre del/de la PRA:	Ferran Prados Carrasco
Fecha de entrega:	02/2023
Titulación o programa:	Máster en Ciencia de Datos
Área del Trabajo Final:	Area Medicina (TFM-Med)
Language:	English
Keywords	Deep Learning, Brain MRI, Variational Autoencoder

Contents

List of Figures

List of Tables

Abstract

Artificial Intelligence is set to be key technology at medicine projects in where models will help doctors at diagnostics with image processing. Magnetic Resonance Images allow doctors to scan brains and create a images that can be used for training models. However, it is too expensive and slow to be able to create large dataset required for models to work at highest accuracy and this project aims to create artificial brain MR images that would enlarge the base dataset combining with real images.

A potential use case in where AI-generated images could be used are anomaly detection by comparing an input image with healthy images to detect if it presents any anomaly to be analyzed by a doctor.

During this project Variational Autoencoders will be used in order to create new images where different existing network architectures will be analyzed before working on the performance tuning with the one which brings better results at initial analysis.

The duration of the project implementation will start on October 24th and will end up by December 25th with one Data Scientist working on the project with a budget of 300 hours that will be distributed during that time frame and that will require GPU computation for processing the different training and testing networks.

This project aims to be a Proof of Concept and the resulting images will be visually analyzed and compared to real images to check if the network is able to generate images that could be used as valid images.

Chapter 1

Introduction

1.1 Context and project justification

Artificial Intelligence has arrived to change the world in almost (if not all) any field. Today, we are surrounded by (and we are using many) AI products like smartphones' face recognition capabilities, home cleaning robots or cars with autopilot options.

From the different AI fields, computer vision is maybe the most popular one and the one which is usually used for explaining AI capabilities to general public. Identifying a cat in a picture could perfectly be the example used in every AI presentation to welcome people to AI.

Image processing is intuitively matched with medical diagnosis by anyone having or not any expertise on the field. Almost every single citizen will have heard of magnetic resonance imaging, and anyone easily transposes 'cat detection' to 'anomaly detection', being the anomaly a tumor or anything else.

There are different techniques to scan people and create images for clinical diagnosis like X-Ray or Magnetic Resonance Imaging. In this project, we will work with Magnetic Resonance Images (MRI) which are images created by a machine with a large bore that scans people lying inside it. The MR technique is non-invasive, it produces no ionizing radiation, and is used to scan almost any part of the body from which this thesis will focus on brain images.

Combining AI diagnosis capabilities on image processing and MRI images, you can think of helping doctors to identify the presence of anomalies or looking for concrete diagnosis for a specific disease.

Obviously, these projects are not easy at all and they face a lot of challenges. One of the first challenges that such AI project faces is the difficulty to obtain a large set of brain images that are needed to train an accurate AI model. Scanning people is too costly and requires a lot of time and the challenge of having a large dataset gets harder once we understand that brain images may differ depending on age or gender.

Today, there is a clear limitation on how to reproduce or obtain healthy brain images for AI-based diagnosis projects while the appearance of new AI techniques known as Autoencoders and Variational Autoencoders introduces a new area of investigation to mitigate the gap.

This project aims to be a proof of concept to create AI-created images with new variational

autoencoders that would serve to augment any existing MRI dataset and that will help to improve the accuracy of brain anomaly detection projects, including those which could be used for overall anomaly detection to others more specific which would help on concrete disease diagnosis.

When submitted, Dr Baris Kanber's thesis proposal was intended to use Autoencoders for MRI reconstruction of images. However, some alternatives were propopsed by him during the first presentation meeting to go beyond that. Dr Baris Kanber suggested to use Variational Autoencoders first, then varying the project objective to create new artificial images instead of recreating them by using the latent space probability distributions. Both proposals were taken with even when more research and experiments would be needed due to the lack of previous projects using this technique.

Variational Autoencoders are seen as an evolution of Autoencoders which store discrete values of each attribute in the latent space while VAEs uses probability distributions. This will be described deeper later in this document.

Personal motivation comes from various angles:

- One is to prove myself that I can work with new AI architectures demonstrating that I have acquired the knowledge needed (deep enough) to be productive and to be able to innovate in the health sector.
- Deep Learning has been the subject which I enjoyed the most, hence continuing with Autoencoders seems natural to me as the next step on AI adoption
- At no doubts, if I can contribute to help on brain issue detection or diagnosis, I will feel my life been completely fulfilled

1.2 Aim of the project

The aim of this project is to serve as a Proof of Concept on how MRI brain images can be artificially generated with Variational Autoencoders which ultimately would serve to enhance existing or new datasets to improve model accuracy by having bigger samples of data

Foreseen projects objectives are:

- Obtain basic knowledge about MRI images and the NIFTI file format
- Obtain and visualise 2-D images from 3-D images in the dataset
- Select what range of slices (2-D images) from brain to be created
- Test different existing networks and choose the one to be used
- Tune network parameters
- Compare generated brain images against real ones, qualitatively

1.3 Project Plan

1.3.1 Resources

1. 1 Data Scientist: Miguel Tablado will be playing this role and will dedicate 300h
2. 1 Coach/Tutor: Baris Kanber will be supervising Miguel Tablado during the project
3. MRI images and demographic information from IXI Dataset
4. GPU resources are needed to train and test the network

1.3.2 High Level Plan

The plan will be executed in 3 different phases with the listed tasks:

1. Phase 1: Analysis
   (a) Gain knowledge on MRI and NIFTI protocol
   (b) Describe images and dataset
   (c) Extract 2D images
   (d) Pre-processing images
2. Phase 2: MR Images creation
   (a) Test different Network architectures
   (b) Tune-up architectural network
3. Phase 3: Project documentation
   (a) Write conclusions
   (b) Create project documentation
   (c) Create project presentation

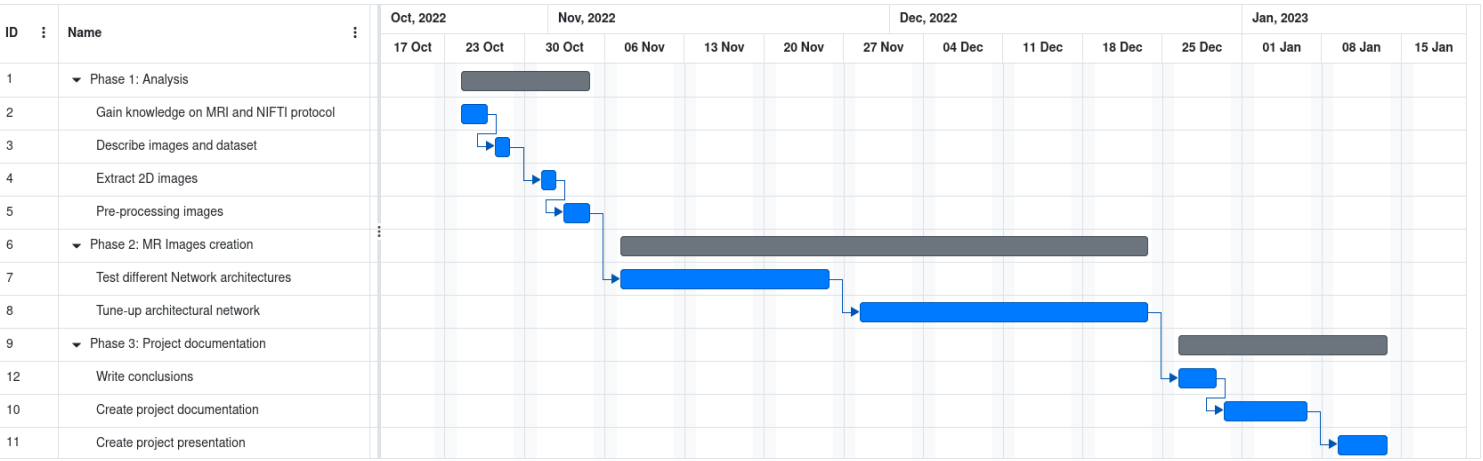


Figure 1.1: Project Plan. Created using: https://www.onlinegantt.com

1.3.3 Tasks

Phase 1: Analysis

During this phase the different tasks will be executed to prepare the work and includes:

1. Gain knowledge on MRI and the NIFTI file format: This activity consists of reading papers and documents to gain sufficient knowledge to execute the project. There is no need to become an expert on the matter but understanding how those files are and how to process them.

2. Describe images and dataset: During this activity, a description of the dataset will be generated with a view of the quality of the dataset for the aim of the project and any findings which could result.

3. Extract 2D images: Load 3D images and extract 2D slices from the original dataset, which will be depicted with code.

4. Pre-processing images: Decide which transformations on the 2D images would help the project like pixel changes or applying gray-scale transformations.

Phase 2: MR Images creation

1. Test different Network architectures: This task will take few existing network architectures and be tested with the dataset so that one of them will be selected to be improved and used as the project architecture.

2. Tune-up architectural network: Tune the selected architecture with useful techniques like changing network layers

Chapter 2

State of the art: related works

2.1 Overview

Since our project is based on brain anomaly detection with variational autoenconders (VAEs), in this section, the state of the art will be described with focus on previous existing anomaly detection projects and variational autoencoders.

Anomaly detection with VAEs has been used in several areas such as fraud detection, KPIs in web applications and, of course, MRI-based brain diagnosis.

In this section, the state of the art is covered by describing VAEs and MRI datasets first, and describing the related works later.

2.2 Variational Autoencoders

The history of Variational Autoencoders could be summarised by sequentially looking into the following papers:

- An and Cho, 2015. Variational autoencoder based anomaly detection using reconstruction probability. [1]
- Xu et al., 2018. Unsupervised anomaly detection via variational autoencoder for seasonal KPIs in web applications [13]
- Zimmerer et al., 2019. Unsupervised anomaly localization using variational auto-encoders. [14]

The first paper must be used to understand the differences between Autoencoders and Variational Autoencoders. This is important since VAE can be interpreted as a new technique with origin on Autoencoders, thus, sharing fundamentals and architectures.

An and Cho [1], described a VAE as

> "A probabilistic graphical model that combines variational inference with deep learning. Because VAE reduces dimensions in a probabilistically sound way, theoretical foundations

are firm. The advantage of a VAE over an autoencoder and a PCA is that it provides a probability measure rather than a reconstruction error as an anomaly score, which we will call the reconstruction probability. Probabilities are more principled and objective than reconstruction errors and does not require model specific thresholds for judging anomalies."

In other words, Jeremy Jordan [9] describes the same in simpler words

"A variational autoencoder provides a probabilistic manner for describing an observation in latent space. Thus, rather than building an encoder which outputs a single value to describe each latent state attribute, we'll formulate our encoder to describe a probability distribution for each latent attribute."

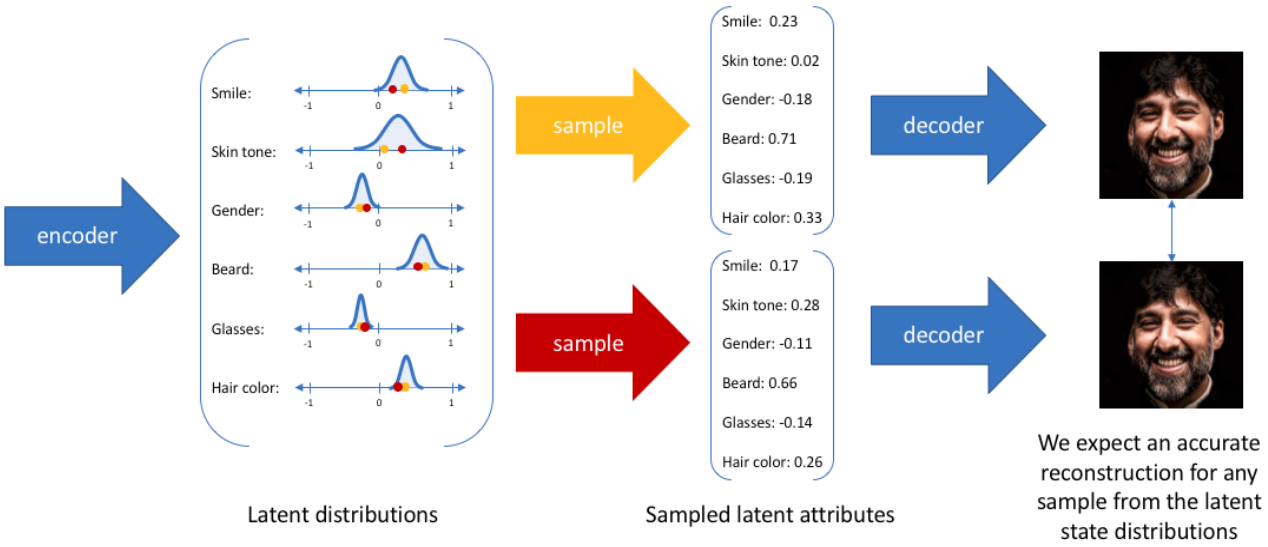


Figure 2.1: Variational Autoencoders by Jerermy Jordan [9]

An and Cho [1] conclude "To summarize, it is the distribution parameters that are being modeled in the VAE, not the value itself."

To understand the differences between Autoencoders and VAEs, An and Cho [1] listed 3 differences:

1. VAE latent space are stochastic variables, not deterministic ones

2. Reconstruction in VAE adds the variability of the reconstruction by considering the variance parameter of the distribution function.

3. Autoencoders use reconstruction errors as anomaly score, while VAE reconstructions are probability measures which work best for heterogeneous data and deciding the reconstruction probability threshold is easier and more objective than deciding for reconstruction error.

Finally, An and Cho [1] proved that VAE outperformed autoencoders and PCA based methods on 2 different popular datasets, MNIST [10] and KDD [8].

Xu et al. later proved the same on a large internet company dataset that shows a combination of seasonal patterns with local variations and the statistics of the Gaussian noises.

The project consisted of detecting anomalies on KPIs on a website; KPIs are time series data measuring metrics such as number of connected users, transactions or orders. Websites for retail industry follow a seasonal distribution and a local variation that explains the increasing trend over days.

Finally, Zimmerer et al. [14], 2018 proved that VAEs could keep the assumption-free principle by improving the used network architecture with a combination of the reconstruction term with the density-based anomaly scoring. Before, the VAEs had shown great potential on unsupervised learning of data distributions but required modifications on the model architecture to perform well for the problem seen during the evaluation, which breaks the assumption-free principle.

In detail, VAE can approximate data distributions by optimizing a lower bound, often termed evidence lower bound (ELBO) which is usually employed as a proxy to compare the likelihood. ELBO is a combination of the reconstruction error and the Kullback-Leibler (KL)-divergence, a backpropagation mechanism. This combination is called a context-encoder VAE or ceVAE.

2.3 MR Images and datasets

The IXI dataset is published at brain-development.org website from Imperial College London and it is the chosen dataset for this project. IXI dataset is a collection of 600 Brain MR Images, in 3-D, collected from 3 different hospitals in London. The images are scanned in 1.5 and 3 Tesla which is the range of the quality that we could usually find in hospitals around the world. Scanners with higher quality (7T) are very rare in the world and so working with this dataset makes sense if you plan to export your work to real production environments.

The images are stored as volumes or 3-D images which require an intense computational effort to process, in this project we will work with 2-D images, or brain slices, which will allow us to scale the number of samples that our model will work with.

Other datasets exist on the internet which can be used for this project, and of course, much more private ones are expected to exist. We will cover a few of them to show the state of the art in this area.

The one sponsored by Facebook AI and hosted at NYU Langone Health is called fastMRI and includes knees and brain MRIs. Nothing better than its overview to describe the aim of this dataset "fastMRI is a collaborative research project from Facebook AI Research (FAIR) and NYU Langone Health to investigate the use of AI to make MRI scans faster".

The dataset consists of 6.970 fully sampled MRI brain images obtained in 3 and 1.5 Tesla magnets like the IXI dataset and includes T1, T2 and FLAIR images.

We can also find Brain Tumor Segmentation (BraTS) challenge [7] which encloses two different tasks of brain tumor detection and classification. The challenge is 10+ years old and as of 2021

provided 2.000 cases with 8.000 MRI scans. Like fastMRI, BraTS provides T1, T2 and FLAIR images in NIFTI files.

BraTS dataset has been pre-processed prior to its release, i.e., co-registered to the same anatomical template, interpolated to the same resolution and skull-stripped.

In conclusion, IXI dataset, which includes a spreadsheet of democratization of the dataset, could be enlarged with BraTS or fastMRI for further research (with the corresponding pre-process alignment).

Since the project aims to create artificial images that could be used with real images to train specific models, using them with different datasets would be interesting for validating the portability of the created images to different hospitals.

2.4 Related Works

In July 2021, Jorge Cardoso published an article [3] describing how 3-D MRI images were created using AI models for diagnosis and prediction of diseases on brains by using a supercomputer called Cambridge-1 and Vector-Quantized Variational Autoencoder (VQ-VAE)

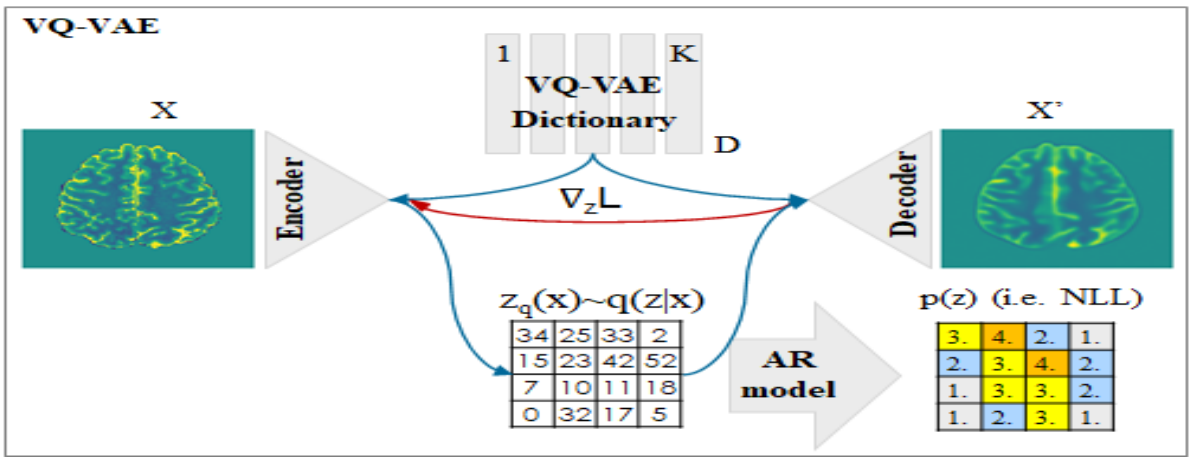


Figure 2.2: VQ-VAE Proposed Approach by Marimont and Tarroni

More recently, In September 2022, Soumick et al. [4] published a paper on how a pipeline (called StRegA) with specific pre-processing and post-processing functions outperformed the Context-Encoder Variational Autoencoder (ceVAE) architectures on unsupervised anomaly detection in brain MRIs.

While Cardoso describes how a supercomputer with 100 GPUs (and other high-quality features) was employed to create MRI brain images across ages, genders and diseases which allows to not only diagnose but to predict diseases, StRegA describes how adding pre and post processing steps to the network architecture on regular computers improves results compared to bare VAE network architectures.

Soumick et al. [4] explains in detail why the unsupervised learning is needed in the field by mainly comparing with supervised learning and by emphasizing its properties. The paper highlights the following list of reasons to justify the usage unsupervised learning:

- Supervised learning relies on the quality and quantity of data, ranging from thousands to millions of labeled data which is a laborious manual annotation work.
- Supervised learning is often focused on one type of pathology
- Supervised learning has demonstrated to work well on concrete anomalies detection with simulated data but many perform poorly with clinical data
- Supervised learning solutions are challenging to develop on some abnormalities such as small vessel disease since the damage is complex on lesion size, contrast, or morphology
- A wide variety of abnormalities can be present in human brain MRI (even simultaneously) which renders a representation of all possible anomalies very challenging (including labeling)
- Anomaly detection is an approach that distinguishes anomalies completely based on characteristics that describe regular data.
- Unsupervised Anomaly Detection is useful for anomalies detection when their manifestation is not known or are very rare; making it difficult to create a large dataset for supervised learning

The StRegA pipeline used for unsupervised anomaly detection used a compact version of the ceVAE (cceVAE) combined with pre and post processors (creating a pipeline) which outperformed VAEs working alone. However, those achievements are only relevant to this project since they have been worked with the same MRI brain images (on a different dataset, BraTS vs IXI), hence it demonstrates that VAEs are optimal for MRI Brain images processing.

In conclusion, the state-of-the-art shows that VAEs have been implemented in combination with supercomputers to create models for diagnosis and prediction of brain diseases using 3-D MRI datasets and that the AI community is looking for cheaper and affordable implementations by innovating new VAE architectures or combining VAEs with pre and post processing actions.

Chapter 3

Project Design and Implementation

3.1 Overview

In this chapter the Design and Implementation tasks are described. They are Phases 1 and 2 of the project plan 1.1 which suffered from some minor deviations and changes mainly after implementing the first VAE model.

In the following sections 2 different VAE implementations will be shown and discussed along with different encoders and decoders architectures which we will be referring to as the network architectures. Before that, the MR Images dataset and NIFTI protocol anayisis is explained with deepest detail on MR Images comprehension and preprocessing.

One key aspect is that in this project there is no validation option for the implemented models rather than the visual analysis of the created images which means that there will be no comparing tables or scorings along this chapter to reflect the quality of the model. However, some training loss data will be used as the reference to partially demonstrate the quality of the different trained models of the different architectures and parameters.

Project implementation consists of TensorFlow Notebooks executed in Google Colabs with standard GPU resources. The source code is stored at GitHub in mtablado/uoc2022_tfm repository which is of public access while IXI dataset and pre-processed images are stored at Google Drive.

The Figure 3.1 shows a diagram with the toolset and how they interact.

The project consists of 3 notebooks:

- MRI which reads, processes and stores IXI images
- VAE which implements VAE model which uses pre-processed images
- VQ-VAE which implements VQ-VAE model which uses pre-processed images

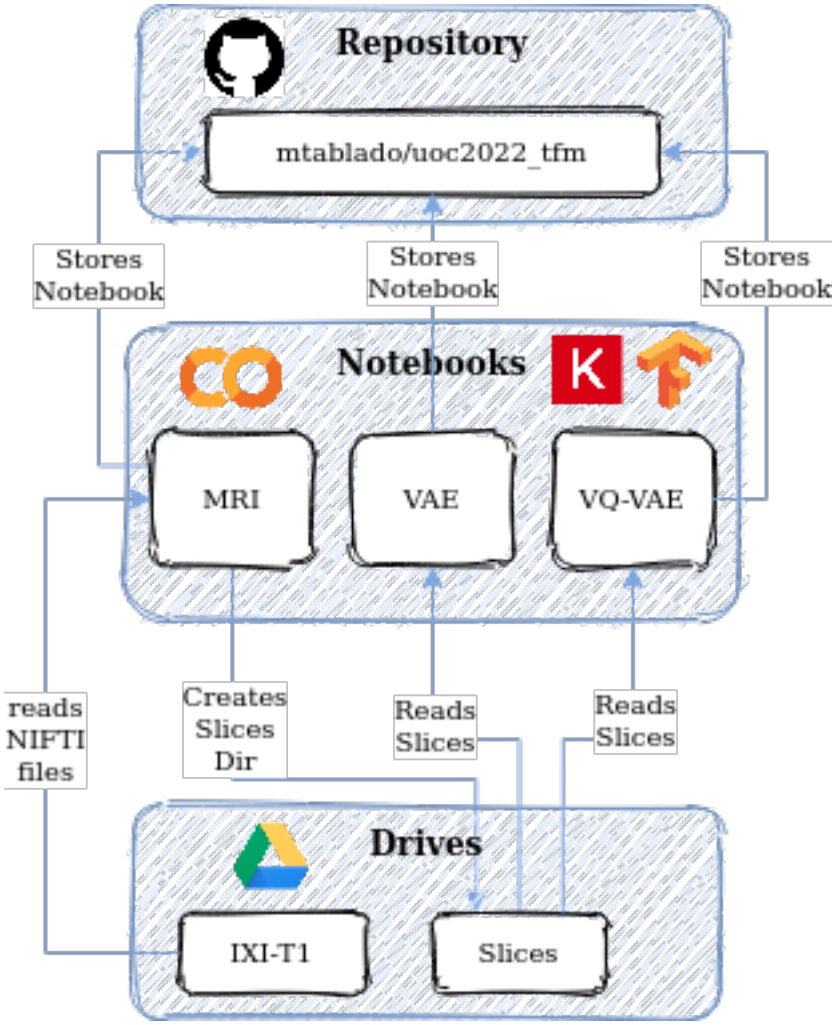


Figure 3.1: Environment and flows of information

3.2 Phase 1: Analysis of MRI images dataset

NIFTI protocol

The NIFTI protocol stores the MRI scan images in 3D format with spatial information that allows the doctors to analyze images knowing the orientation along with some other metadata. Data is stored as vectors with voxel information, so the voxel is stored with its magnitude and its direction.

This project is using the nibabel [2] library to load and process MRI images from NIFTI files. The library loads NIFTI files and retrieves an in-memory object with the image data containing slices in a 3-dimensional array. Accessing slices is then as easy as using the three indexes in the array, where you can easily get a slice by using the slice number from your chosen axis.

Loading images

Extract 2D images: Load 3D images and extract 2D slices from the original dataset, which will be depicted with code.

After loading NIFTI images with the nibabel library ...

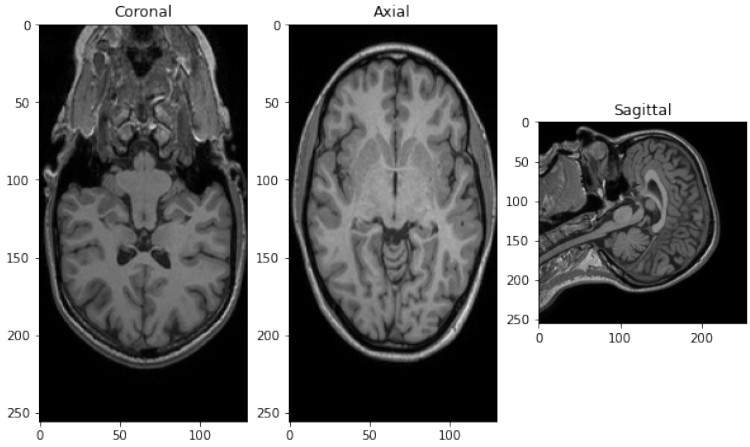


Figure 3.2: Coronal, Axial and Sagittal images

Images are loaded in the following shape:

- 256 slices of Coronal or trasversal plane
- 256 slices of Axial or horizontal plane
- 130 slices of Sagittal

Deciding what slices to use

Looking at the three different slices in Figure 3.2, it is obvious that images have not been processed to show only the brain but the whole skull. The aim of the project is to create artificial brain images, not head ones. A model that would learn head images would require much more time and resources than for learning just the brain (which is not an easy task at all either).

While observing the BraTS [7] challenge dataset and project description, the skull stripping concept arise. Skull stripping consists on removing anything that does not pertain to brain or, in other words, extract the brain from the skull.

This project cannot afford such implementation within the project scope. So, when deciding what slices to use, it became the only decision factor. It happened to be that the rest of considerations where just debatable and clearly much less important.

In conclusion, the noise that bones, eyes and the rest of the items visible in coronal and sagittal axis look very high compared to the axial slices in where, properly selected, slices will only show a surrounding skull bone.

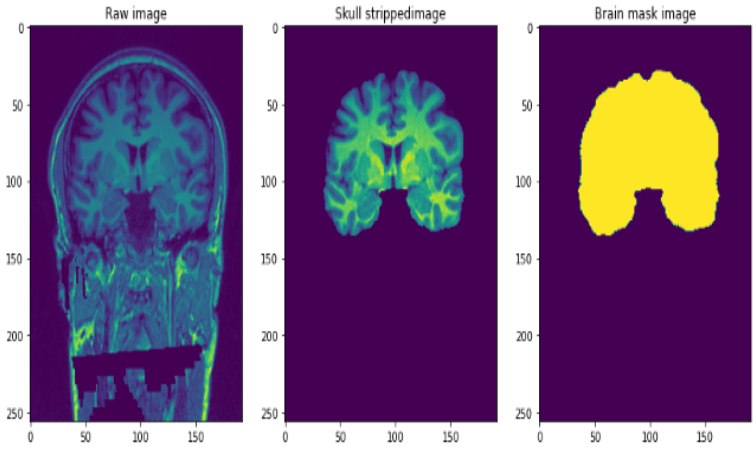


Figure 3.3: Skull stripped image example; Source: Analytics Vidhya [6]

Once that the axial slices were chosen, a sample of brain scans were visually analyzed to find a range of slices that could be used.

Figure 3.4 shows a red line which bottom-up delimits the range that goes from above the eyes to the top which is the range that excludes as much non-brain pixels as possible

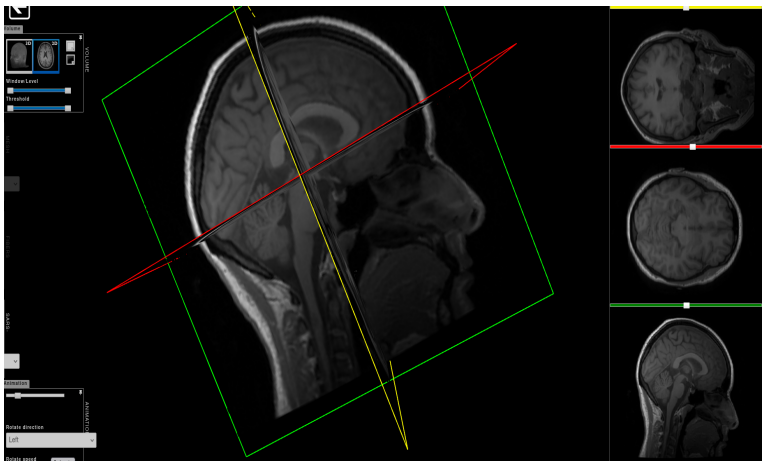


Figure 3.4: In red, slice 160 as bottom delimiter

Also, the slices closer to the top were also discarded to avoid higher heterogeneity on brain images (size and content). See Figure 3.5.

It can be interpreted that 3D images are in use since 30 slices of each brain are used which constitue a 3D section of a brain. But, similarly, it could be interpreted that each and every slice of those 30 could happen to be data augmentation or different people with very similar brains.

Data Pre-processing and Data Augmentation

Different pre-processing and data augmentation techniques were briefly evaluated to decide wheter if they would improve the model's results or not. Table 3.1 shows the decisions made on each evaluated

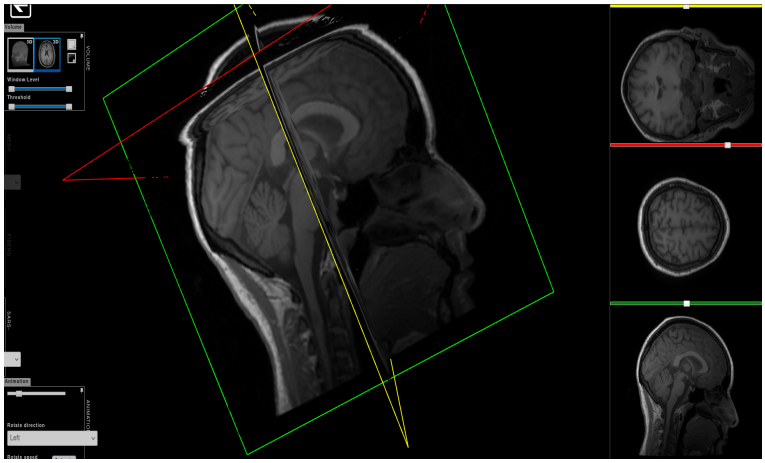


Figure 3.5: In red, slice 190 as top delimiter

technique.

Overall, pre-processing actions that were feasible to implement in a timely manner were approved while those that could risk the project plan were discarded.

Also, data augmentation techniques that could result on unpredictable latent space results were initially discarded to be re-evaluated at a future work when the first results and conclusions would be available.

Technique	Decision	Justification
Gray scale	Approved	Applying gray scale to scans seems a natural choice and is usually applied on image processing
Rescaling	Approved	Rescaling RGB values from 0-256 to 0-1 that can be better handled by the model
Pixeling	Approved	Down-pixeling images from 256 to 128 which requires much less resources and without losing much quality
Shear Intensity	Declined	Slanting images would distort the latent space leading to create distorted artificial images
Shifts and Rotation	Declined	Similar to shear intensity shifting images is seen as a method not accurate for MRI processing as MRI images will not appear shifted in any prediction phase
Add Noise	Declined	Adding noise is interesting for improving results and key to allow model transfering to other datasets obtained with different scanners and their settings or image quality. It was declined due to time constraints.

Table 3.1: Pre-processing techniques decisions table

Storing data

In order to separate image loading from model training into different steps, selected 2D images are stored in PNG format after being pre-processed.

The images are stored in 2 different directories spliting the dataset into 90% for training and 10% for test subsets resulting in 16.268 and 1.162 images respectively.

This process is slow due to the need of intensive disk writing operations. Saving 16k images at Google Drive lasts for about 15 minutes. As a consequence, images were stored with original 256 pixels so that they could be scaled down to 128 or less if needed. Creating different directories with images stored in different sizes would waste around 2.5 Gigabytes per sizing.

3.3 Phase 2: MR Images creation

3.3.1 Variational Autoencoder

The source code of the implemented VAE can be found at vae.ipynb file inside the project repository and it is inspired by Variational Autoencoder [5] keras example.

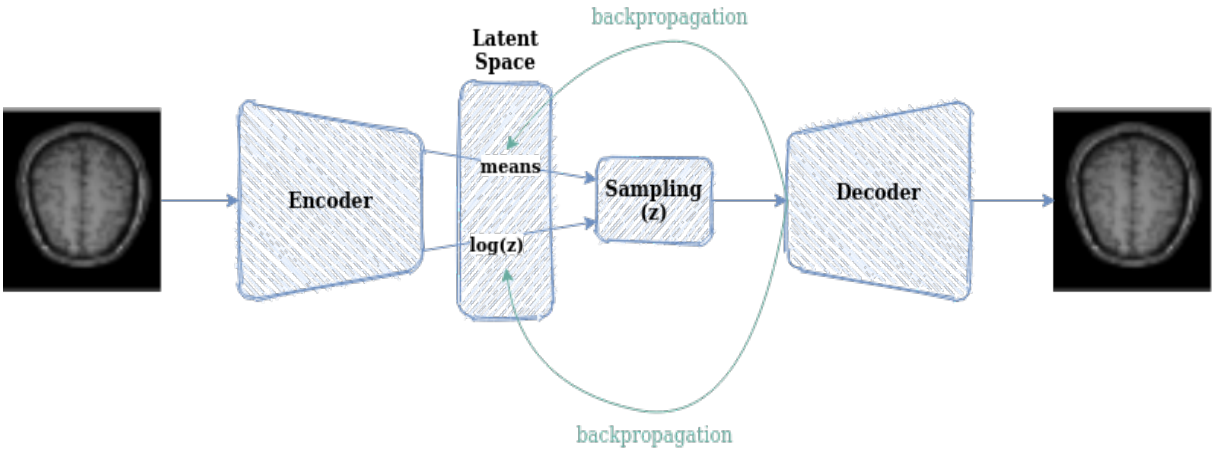


Figure 3.6: Diagram of Implemented VAE Architecture

The encoder consists of a Convolutional Neural Network (CNN) that uses the preprocessed brain slices images as input and then two vectors that form the latent space, which has been sized to store 1024 attributes. CNN implementation is described later in a dedicated section.

Encoder output vectors are:

- A 1024-dimensional vector of means (μ), and
- a 1024-dimensional vector of standard deviations (z) as $log(z)$ to make sure they are positive.

Then those vectors are sampled and fed into the decoder to reconstruct the MRI. Backpropagation is applied to correct latent space issues and let the model be more accurate.

To implement above architecture in Keras, a new model was created by overriding the Keras Model abstract class and implementing the methods that the interface requires. So, specific methods are coded to process each and every train step and the metrics property is overriden to store the training loss, the reconstruction loss and the Kullback-Leibler divergence data.

The training method calculates the reconstruction loss and the KL divergence. Then, their values are aggregated to result in the training loss value.

$$TrainingLoss = ReconstructionLoss + KLDivergence \tag{3.1}$$

The reconstruction loss measures how close the encoder input and the encoder output are by calculating the MSE of the difference between images. On the other hand the KL-Divergence measures the distance between two probability distributions. So, minimizing the KL loss in this case means ensuring that the learnt means and variances are as close as possible to those of the target (normal) distribution.

The latent space of the VAE is usually a Gaussian distribution. Gaussian distribution is also used here since it has very good computational properties.

The sampling process is implemented as an encoder Keras Layer which takes both vectors and samples with a formula that aggregates the mean and the variance which adjusted with a random factor.

The KL-Divergence issue

As introduced in 3.1 overview section, some deviations from the project happened. The KL-Divergence interpretation and calculation was (and remains) a problem. Contrary to what should happen, instead of decreasing, the KL-Divergence metric increases along the epochs.

Several calculation formulas for KL-Divergence were tested with little result changes (they are commented in at source file).

The best model accuracy comes at maximizing the ELBO, which happens when you minimize both reconstruction loss and KLDivergence.

Despite the KL-Divergence issue, the ELBO improves in each step thanks to the higher minimization of the reconstruction loss

$$|ReconstructionLoss_i - ReconstructionLoss_{i-1}| > |KLDivergence_i - KLDivergence_{i-1}| \tag{3.2}$$

This casualty is allowing the model to keep learning.

This issue inclined the project to create a secondary approach, train also a VQ-VAE model to compare results which is describe in the following section, since the time constraint did not allow the project to keep investigating on this issue.

3.3.2 Vector-Quantized Variational Autoencoder

The source code of this section can be found at vq-vae.ipynb file inside the project repository and it is inspired by Vector-Quantized Variational Autoencoder [11] keras example, which code was used as starting point.

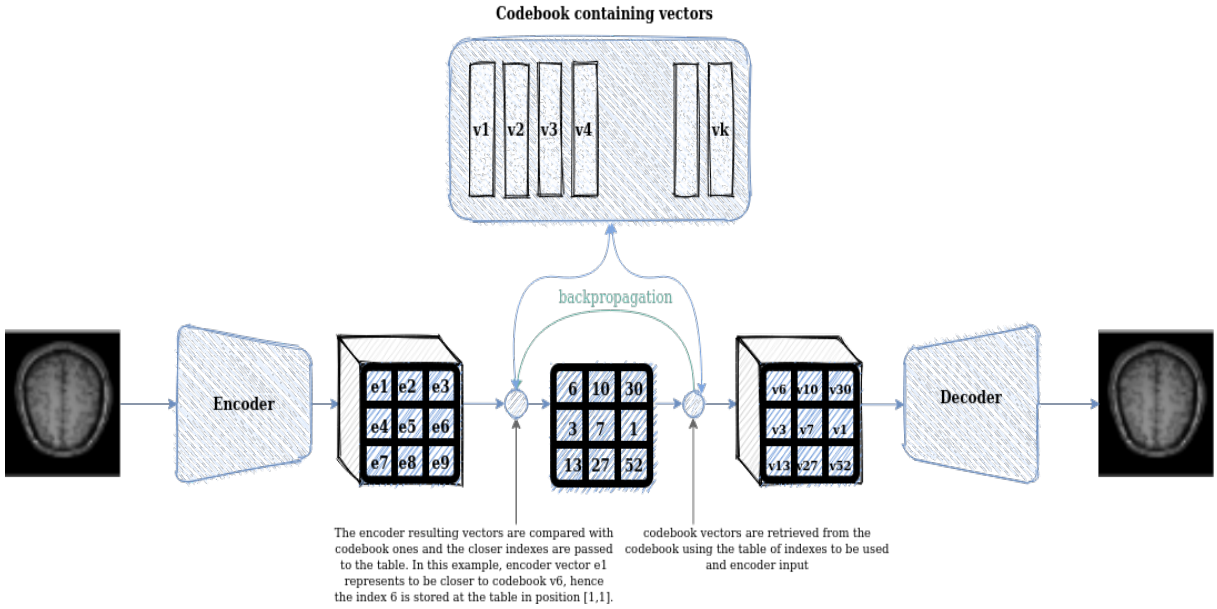


Figure 3.7: Diagram of Implemented VQ-VAE Architecture

As VAE, VQ-VAE use CNN but, in this case, the encoder generates a set of vectors with discrete representations (instead of continuous like VAE) that is During training phase VQ-VAE creates a codebook of vectors which are also learnable and predictable. Once that a new image is processed, the encoder output vector of z is compared with the codebook vectors and the one closer to it is used later as the input of the decoder. Proximity between vectors is calculated with the L2-norm.

The resulting indexes are stored into a set that then serves as the indexing object to build a new set of vectors that will be used as decoder input.

To do above-mentioned steps, a Vector Quantizer object (a new type of Keras Layer) was created to select the quantized vectors from the latent space which are closer to the encoder output vectors and then, those quantized vectors will be used as decoder inputs.

To implement above architecture in Keras, as like for VAEs and along with the Vector Quantized object, a new model was created by overriding the Keras Model abstract class and implementing the methods that the interface requires. So, specific methods are coded to process each and every train step and the metrics property is overriden to store the total loss, the reconstruction loss and the VQ loss.

The training method calculates the reconstruction loss and the VQ loss and uses them to calculate the training loss (total training here) with the following formula:

$$TotalLoss = ReconstructionLoss + \sum(VQlosses) \tag{3.3}$$

3.3.3 CNN Network Architectures

The implemented CNN is based on Context-encoding Variational Autoencoder for Unsupervised Anomaly Detection paper [15] in where Zimmerer et al. describe it as:

> For the encoder and decoder networks, we chose fully convolutional networks with five 2D-Conv-Layers and 2D-Transposed-Conv-Layers respectively with CoordConv, kernel size 4 and stride 2, each layer followed by a Leaky-ReLU non-linearity. The encoder and decoder are symmetric with 16, 64, 256, 1024 feature maps and a latent variable size of 1024

Figure 3.8 shows the encoder model summary output after compiling the model. It is truncated to show only CNN-related layers removing VAE or VQ-VAE specific layers. Full output can be checked at Figures 3.10 and 3.11

Model: "Encoder"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	[(None, 128, 128, 1)]	0	[]
conv_1 (Conv2D)	(None, 64, 64, 16)	272	['input_layer[0][0]']
bn_1 (BatchNormalization)	(None, 64, 64, 16)	64	['conv_1[0][0]']
lrelu_1 (LeakyReLU)	(None, 64, 64, 16)	0	['bn_1[0][0]']
conv_2 (Conv2D)	(None, 32, 32, 32)	8224	['lrelu_1[0][0]']
bn_2 (BatchNormalization)	(None, 32, 32, 32)	128	['conv_2[0][0]']
lrelu_2 (LeakyReLU)	(None, 32, 32, 32)	0	['bn_2[0][0]']
conv_3 (Conv2D)	(None, 16, 16, 64)	32832	['lrelu_2[0][0]']
bn_3 (BatchNormalization)	(None, 16, 16, 64)	256	['conv_3[0][0]']
lrelu_3 (LeakyReLU)	(None, 16, 16, 64)	0	['bn_3[0][0]']
conv_4 (Conv2D)	(None, 8, 8, 256)	262400	['lrelu_3[0][0]']
bn_4 (BatchNormalization)	(None, 8, 8, 256)	1024	['conv_4[0][0]']
lrelu_4 (LeakyReLU)	(None, 8, 8, 256)	0	['bn_4[0][0]']
conv_5 (Conv2D)	(None, 4, 4, 1024)	4195328	['lrelu_4[0][0]']
bn_5 (BatchNormalization)	(None, 4, 4, 1024)	4096	['conv_5[0][0]']
lrelu_5 (LeakyReLU)	(None, 4, 4, 1024)	0	['bn_5[0][0]']
flatten (Flatten)	(None, 16384)	0	['lrelu_5[0][0]']

Figure 3.8: CNN Encoder model summary output

The input size of the CNN is [128,128,1] which are the 128 pixels resulting of the preprocessing downsampling and the usage of 1 channel for the gray scale data.

Figure 3.9 shows the decoder model summary output after compiling the model.

Both of the above-described VAE and VQ-VAE models use this network architecture to encode and decode images where different steps/layers are put in between them to implement the differences of the two models.

Model: "Encoder"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	[(None, 128, 128, 1)]	0	[]
conv_1 (Conv2D)	(None, 64, 64, 16)	272	['input_layer[0][0]']
bn_1 (BatchNormalization)	(None, 64, 64, 16)	64	['conv_1[0][0]']
lrelu_1 (LeakyReLU)	(None, 64, 64, 16)	0	['bn_1[0][0]']
conv_2 (Conv2D)	(None, 32, 32, 32)	8224	['lrelu_1[0][0]']
bn_2 (BatchNormalization)	(None, 32, 32, 32)	128	['conv_2[0][0]']
lrelu_2 (LeakyReLU)	(None, 32, 32, 32)	0	['bn_2[0][0]']
conv_3 (Conv2D)	(None, 16, 16, 64)	32832	['lrelu_2[0][0]']
bn_3 (BatchNormalization)	(None, 16, 16, 64)	256	['conv_3[0][0]']
lrelu_3 (LeakyReLU)	(None, 16, 16, 64)	0	['bn_3[0][0]']
conv_4 (Conv2D)	(None, 8, 8, 256)	262400	['lrelu_3[0][0]']
bn_4 (BatchNormalization)	(None, 8, 8, 256)	1024	['conv_4[0][0]']
lrelu_4 (LeakyReLU)	(None, 8, 8, 256)	0	['bn_4[0][0]']
conv_5 (Conv2D)	(None, 4, 4, 1024)	4195328	['lrelu_4[0][0]']
bn_5 (BatchNormalization)	(None, 4, 4, 1024)	4096	['conv_5[0][0]']
lrelu_5 (LeakyReLU)	(None, 4, 4, 1024)	0	['bn_5[0][0]']
flatten (Flatten)	(None, 16384)	0	['lrelu_5[0][0]']

Figure 3.9: CNN Decoder model summary output

3.3.3.1 Hidden Layers

As described in this section, the implemented CNN is based on Context-encoding Variational Autoencoder for Unsupervised Anomaly Detection paper [15] in where Zimmerer et al. describe it as:

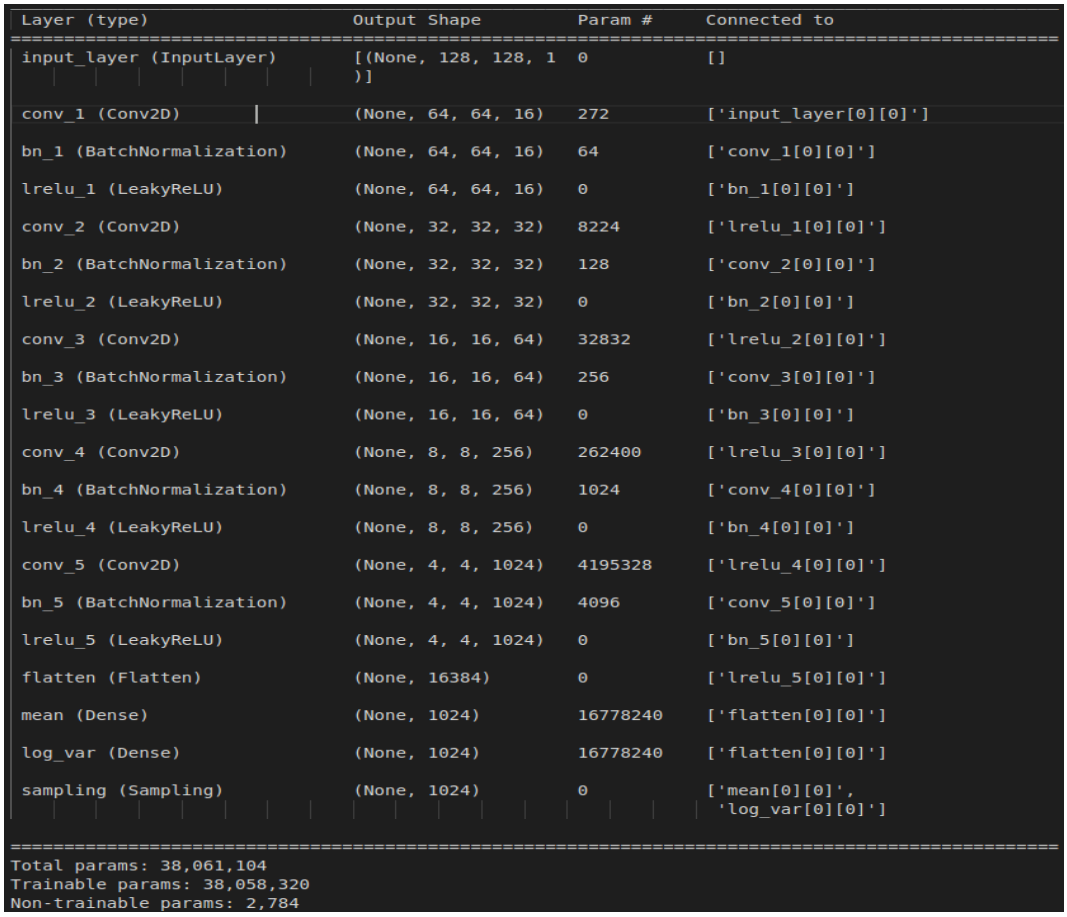
> For the encoder and decoder networks, we chose fully convolutional networks with five 2D-Conv-Layers and 2D-Transposed-Conv-Layers respectively with CoordConv, kernel size 4 and stride 2, each layer followed by a Leaky-ReLU non-linearity. The encoder and decoder are symmetric with 16, 64, 256, 1024 feature maps and a latent variable size of 1024

In project's Convolutional Neural Network (CNN), a Normalization layer is added between them resulting in the following composite:

- Convolutional neuron (Conv2D or Conv2DTranspose)
- Normalization neuron (BatchNormalization)
- ReLu neuron (LeakyReLu)

The convolutional layer is paremetrized with 4 kernels, stride 2 and padding 0. The kernel size aims to avoid the network learning too fast, while using stride 2 to lower the training resources needed and train faster.

According to Martin Riva [12] "Batch Norm is a normalization technique done between the layers of a Neural Network instead of in the raw data. It is done along mini-batches instead of the full data set. It serves to speed up training and use higher learning rates, making learning easier. The standard



Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	[(None, 128, 128, 1)]	0	[]
conv_1 (Conv2D)	(None, 64, 64, 16)	272	['input_layer[0][0]']
bn_1 (BatchNormalization)	(None, 64, 64, 16)	64	['conv_1[0][0]']
lrelu_1 (LeakyReLU)	(None, 64, 64, 16)	0	['bn_1[0][0]']
conv_2 (Conv2D)	(None, 32, 32, 32)	8224	['lrelu_1[0][0]']
bn_2 (BatchNormalization)	(None, 32, 32, 32)	128	['conv_2[0][0]']
lrelu_2 (LeakyReLU)	(None, 32, 32, 32)	0	['bn_2[0][0]']
conv_3 (Conv2D)	(None, 16, 16, 64)	32832	['lrelu_2[0][0]']
bn_3 (BatchNormalization)	(None, 16, 16, 64)	256	['conv_3[0][0]']
lrelu_3 (LeakyReLU)	(None, 16, 16, 64)	0	['bn_3[0][0]']
conv_4 (Conv2D)	(None, 8, 8, 256)	262400	['lrelu_3[0][0]']
bn_4 (BatchNormalization)	(None, 8, 8, 256)	1024	['conv_4[0][0]']
lrelu_4 (LeakyReLU)	(None, 8, 8, 256)	0	['bn_4[0][0]']
conv_5 (Conv2D)	(None, 4, 4, 1024)	4195328	['lrelu_4[0][0]']
bn_5 (BatchNormalization)	(None, 4, 4, 1024)	4096	['conv_5[0][0]']
lrelu_5 (LeakyReLU)	(None, 4, 4, 1024)	0	['bn_5[0][0]']
flatten (Flatten)	(None, 16384)	0	['lrelu_5[0][0]']
mean (Dense)	(None, 1024)	16778240	['flatten[0][0]']
log_var (Dense)	(None, 1024)	16778240	['flatten[0][0]']
sampling (Sampling)	(None, 1024)	0	['mean[0][0]', 'log_var[0][0]']

Total params: 38,061,104
Trainable params: 38,058,320
Non-trainable params: 2,784

Figure 3.10: CNN VAE encoder model summary output

deviation of the neurons' output.". The project is loading raw data in batches which are normalized as a preparation step, however, normalizing data between each layer might help.

ReLu activation is usually applied in Deep Learning (and so it is in the project) right after every convolutional layer to remove linearity since, basically, all the previous operations are linear.

3.3.4 Flow from directory

When working with large image datasets data cannot be loaded in memory and then passed to the model for training. Instead, data is required to be processed in batches to alleviate the need of compute resources.

To do so, Keras provides the Flow from directory method which allows you to configure data loading parameters and several pre-processing actions.

Once configured, the flow from directory method retrieves an iterator object, which is passed to the model which will invoke it to get the next batch of images when needed during the training steps in each epoch.

Aligning the model batch size with the iterator batch size avoids unnecessary calls and more importantly tunes memory usage while training the models.

```
Model: "encoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 128, 128, 1)]     0

 conv_0 (Conv2D)             (None, 64, 64, 32)        544

 bn_0 (BatchNormalization)   (None, 64, 64, 32)        128

 lrelu_0 (LeakyReLU)         (None, 64, 64, 32)        0

 conv_1 (Conv2D)             (None, 32, 32, 32)        16416

 bn_1 (BatchNormalization)   (None, 32, 32, 32)        128

 lrelu_1 (LeakyReLU)         (None, 32, 32, 32)        0

 conv_2 (Conv2D)             (None, 16, 16, 64)        32832

 bn_2 (BatchNormalization)   (None, 16, 16, 64)        256

 lrelu_2 (LeakyReLU)         (None, 16, 16, 64)        0

 conv_3 (Conv2D)             (None, 8, 8, 64)          65600

 bn_3 (BatchNormalization)   (None, 8, 8, 64)          256

 lrelu_3 (LeakyReLU)         (None, 8, 8, 64)          0

 conv_4 (Conv2D)             (None, 4, 4, 64)          65600

 bn_4 (BatchNormalization)   (None, 4, 4, 64)          256

 lrelu_4 (LeakyReLU)         (None, 4, 4, 64)          0

 conv_5 (Conv2D)             (None, 2, 2, 64)          65600

 bn_5 (BatchNormalization)   (None, 2, 2, 64)          256

 lrelu_5 (LeakyReLU)         (None, 2, 2, 64)          0

 conv2d_1 (Conv2D)           (None, 2, 2, 16)          1040

=================================================================
Total params: 248,912
Trainable params: 248,272
Non-trainable params: 640
```

Figure 3.11: CNN VQ-VAE encoder model summary output

The method was paremetrized to rescale the pixels from original 256 to final 128. Training and testing images were extracted from NIFTI files and stored in PNG format with original size so that they could be resized as desired at this stage. This configuration allowed the project to experiment with both sizes much easier.

3.3.5 Creating artificial images from latent space

To create artificial images from the latent space the decoder input must be generated. As explained, the decoder expects an object from the latent space, which means that 1024 values are expected in the normal Gaussian distribution. This is randomly generated with NumPy library and its numpy.random.normal function.

In figure 2.1, samples are created from 6 variables, each of them corresponding to a learnt feature. Two different images are shown created from 2 different samples, yellow and red ones. Yellow and red points shown in the image (inside "Latent distributions") could have perfectly been created as described in the previous paragraph.

In this project, the 1024 generated values are equivalent to those yellow and red points and they define the small variations on brain features that will be used when creating the new image; refer to

diagram 3.12 for a clear representation of the pipeline.

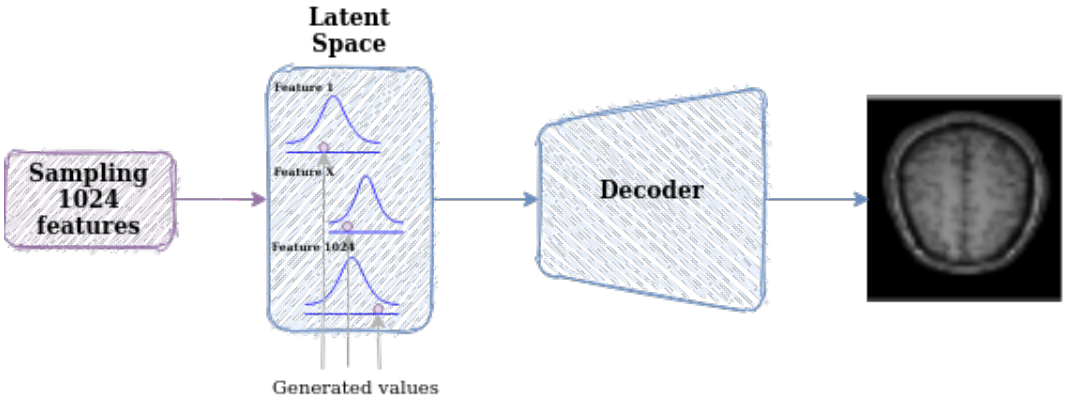


Figure 3.12: Image created from latent space

3.3.6 Image reconstruction

Once that VAE and VQ-VAE models are created, test images are used for testing purposes.

Testing an image process requires to encode and decode it. When encoding an image, the encoder will create a vector of values (z) which will be passed to the decoder to be reconstructed with the formula:

$$z = \mu + e^{(0,5*\sigma)} * random(vector) \tag{3.4}$$

The random vector is generated with the same method used for creating images from latent space (see section 3.3.5). Hence, image reconstruction is very similar to images created from latent space with the difference that the random values are influenced by the means and variance of the model.

3.3.7 Results and Experiments

3.3.7.1 Final Results

The described implemented architecture of VAE and VQ-VAE models results show that the models are learning but they are not able to create images that could be interpreted as real.

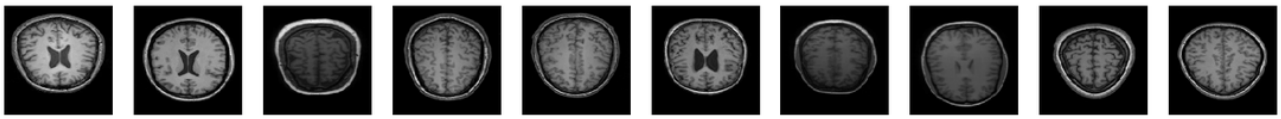


Figure 3.13: Real MR Images

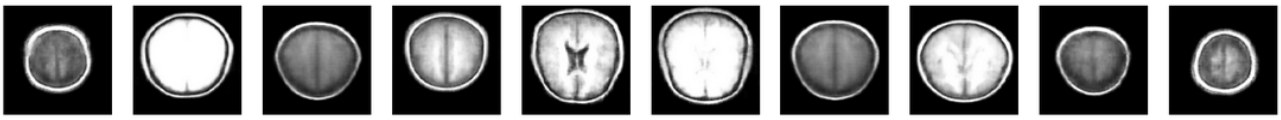


Figure 3.14: VAE Reconstructed Images

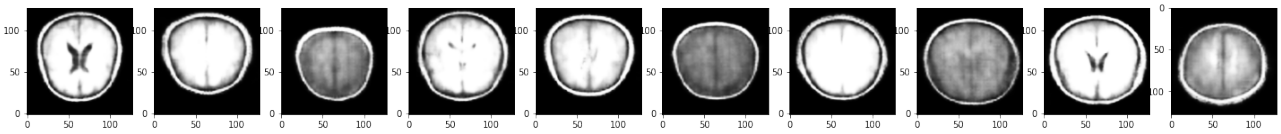


Figure 3.15: VQ-VAE Reconstructed Images

Reconstructed images are slices that were created using the test data. They were created using the output of the encoder as the decoder input not artificially generated.

Note that real images shown in Figure 3.13 do correlate with images shown in Figures 3.14 and 3.15.

Looking at technical results, the models seem to learn very slowly but they keep learning even at 500 epochs so the models' limits are yet to be discovered. On the other hand, as explained in dedicated section, VAE model shows an issue with the KL Divergence, it becomes higher over epochs while it must be decreased.

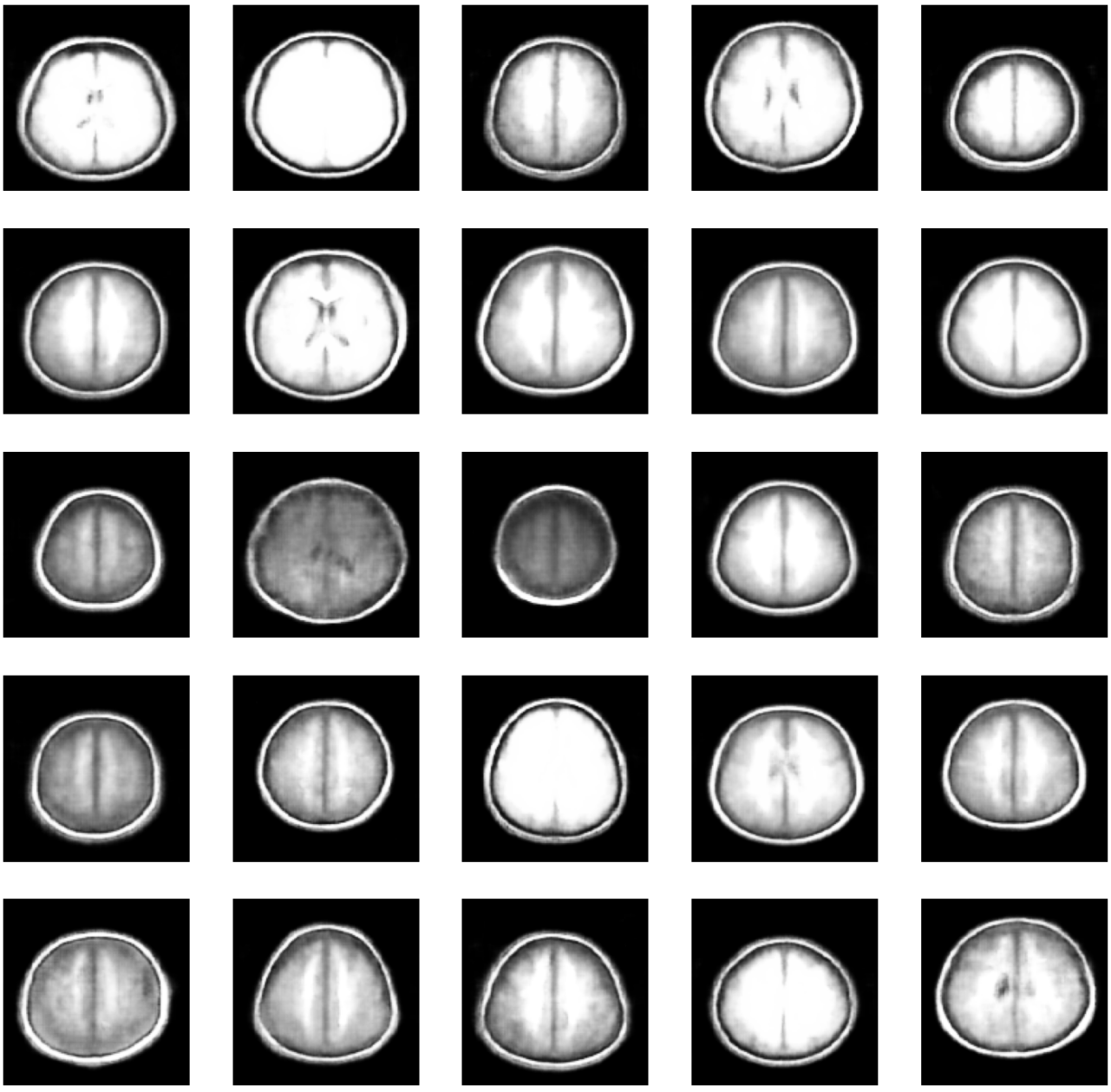


Figure 3.16: Artificial VAE images created from latent space

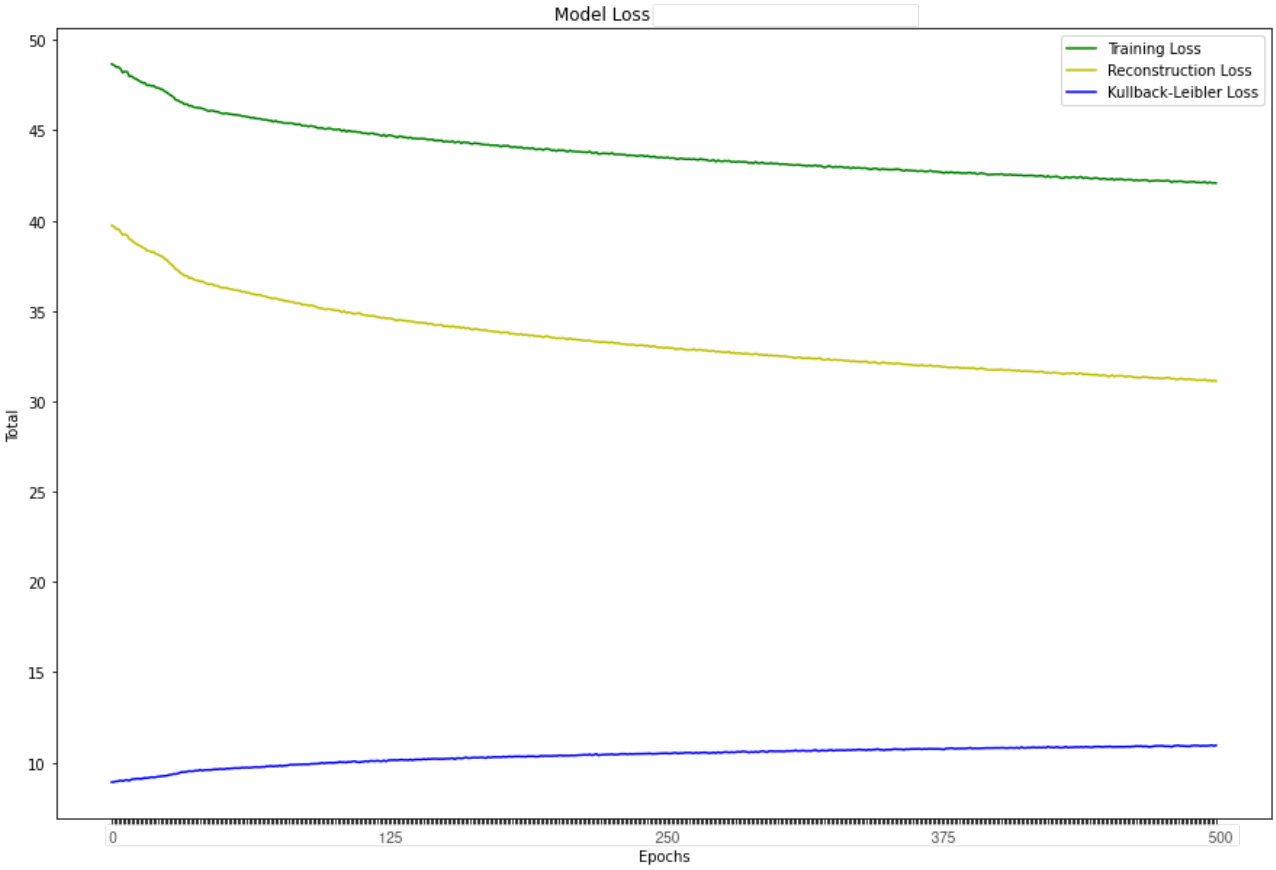


Figure 3.17: VAE Training loss

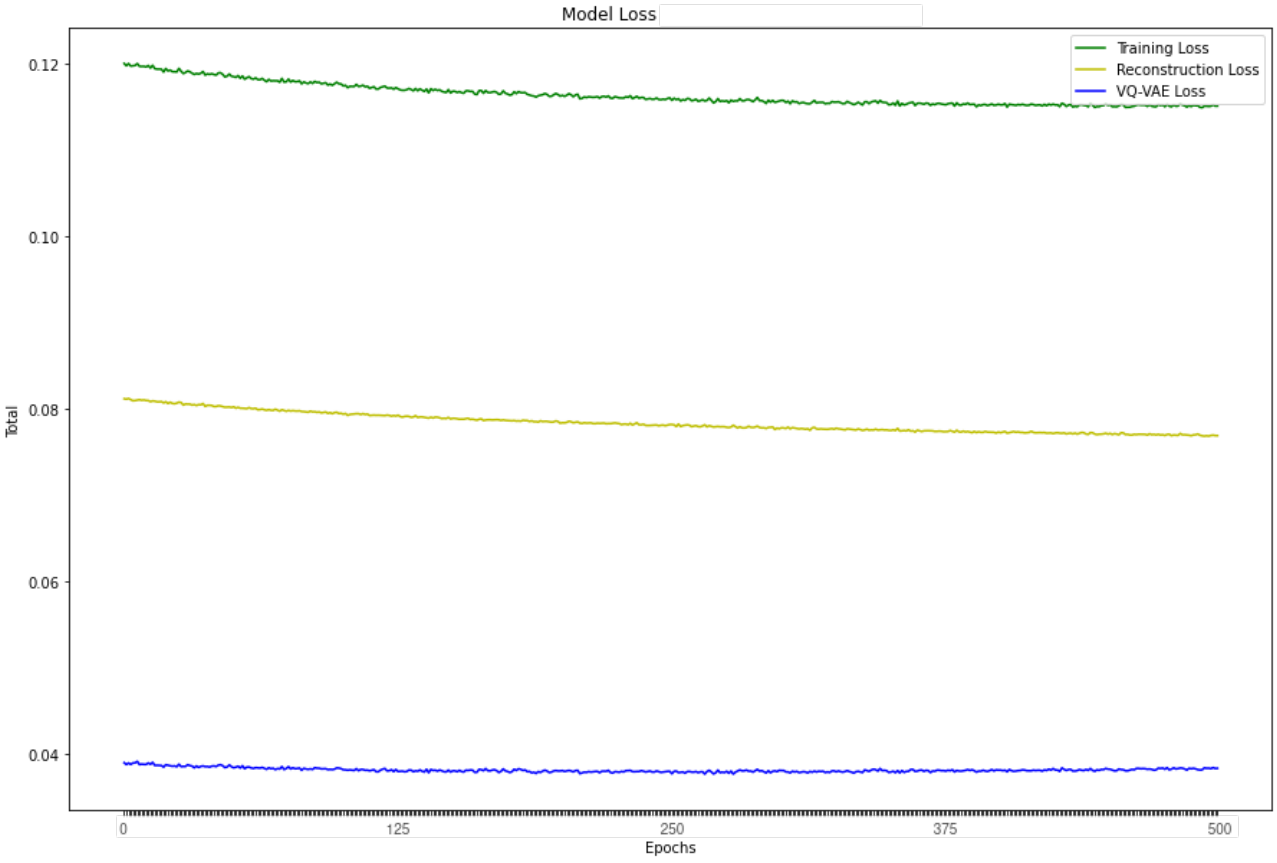


Figure 3.18: VQ-VAE Training loss

3.3.7.2 Experiments

The project implementation methodology followed determined that after every task a fast status analysis was assessed. The reason to do so was the research nature of the project that required continuous evalution on the process and results.

Most of the experiments were executed during the creation of images phase. Whereas the resulting pre-process tasks were overall clear due to MRI slices deep study, executed at the analysis phase of the project, the creation of images required analysis and implementation cycles since at the analysis phase of the project VAEs' theory was checked but the implementation tasks were not.

Instead of creating it from scratch, a simple Convolutional Neural Network (CNN) that is used to show VAE example on MNIST [10] digits was picked. It was only used to test the code functionality only, results were not important and they were discarded.

Once that the functionality was proved to work, the 5-block network was implemented with the following features (details only the ones that vary from the final model):

- 128, 64, 32, 16 and 8 dimensions and a latent space of 200
- kernel size was 3
- trained on 30 epochs only
- 256 pixels

Producing the images shown in Figure 3.19 and losses shown in Figure 3.20

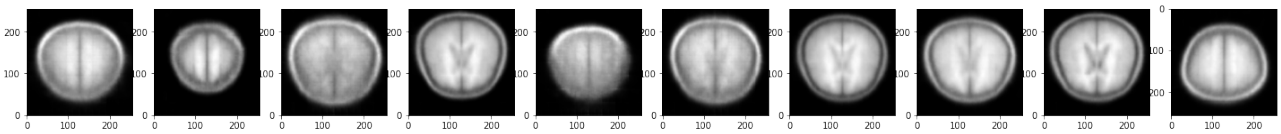


Figure 3.19: VAE Results on experiments I

At that time, since results on images were not ideal the new approach was created in order to double-check if the kl-divergence issue could be the reason for not getting better results. A VQ-VAE implementation was created along with an extra sixth layer added at the begining of the encoder, looking for a slower learning approach. The VQ-VAE implementation with 6 layers produced the images shown in Figure 3.21 and losses shown in Figure 3.22

Then, after checking that VAE and VQ-VAE produced similar results it was decided to look into different angles of potential improvements and the downsampling of pixels experiment was chosen to be next. Along with downsampling, the number of epochs was increased to 200.

VAE Produced the images shown in Figure 3.23 and losses shown in Figure 3.24

After that, some other parameters changes were tested with no remmarkable results compared to the latest results obtained. Tested parameters were learning rates 0.0001, 0.0005 and kernel size 3, trained on 500 epochs.

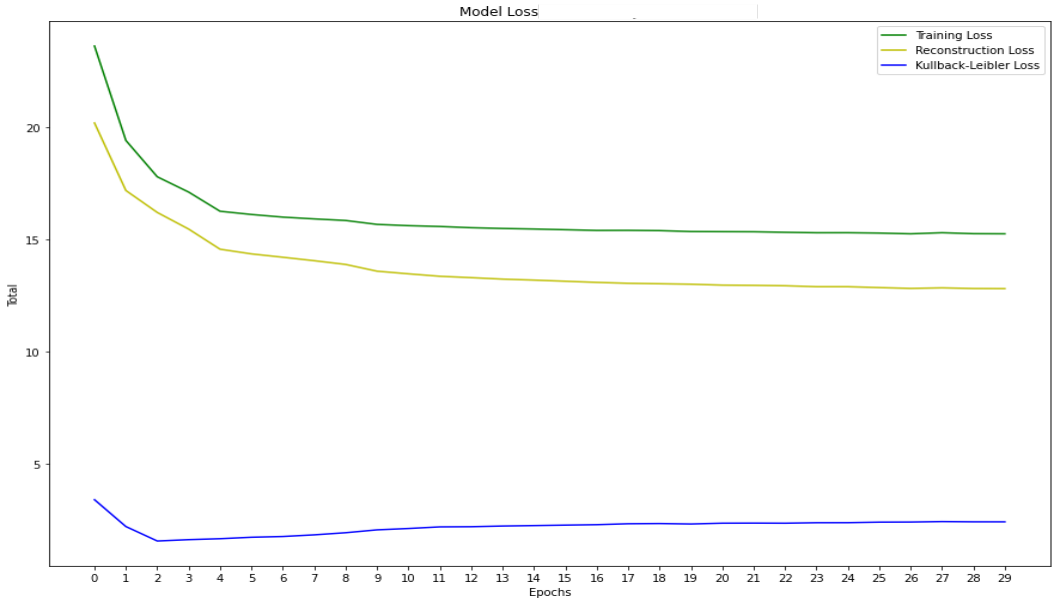


Figure 3.20: VAE Losses on experiments I

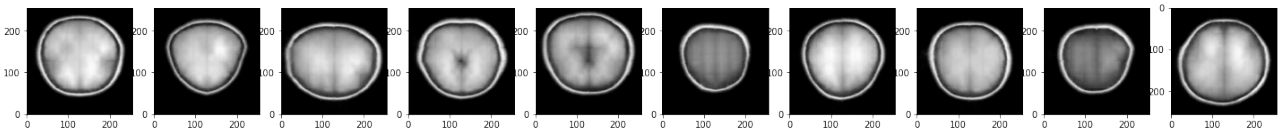


Figure 3.21: VQ-VAE Results on experiments II

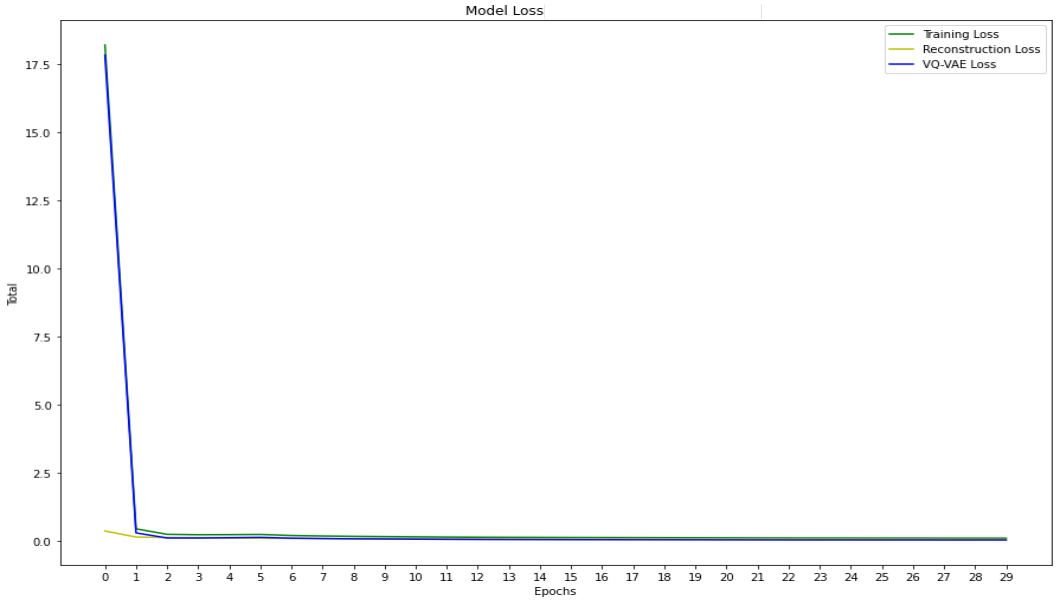


Figure 3.22: VQ-VAE Losses on experiments II

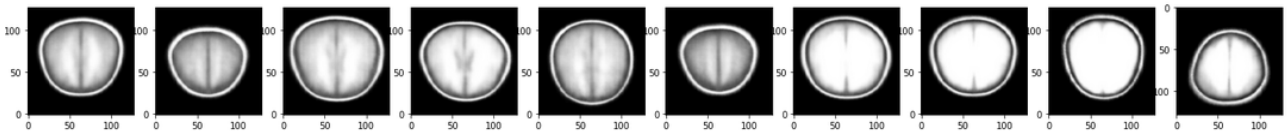


Figure 3.23: VAE Results on experiments III

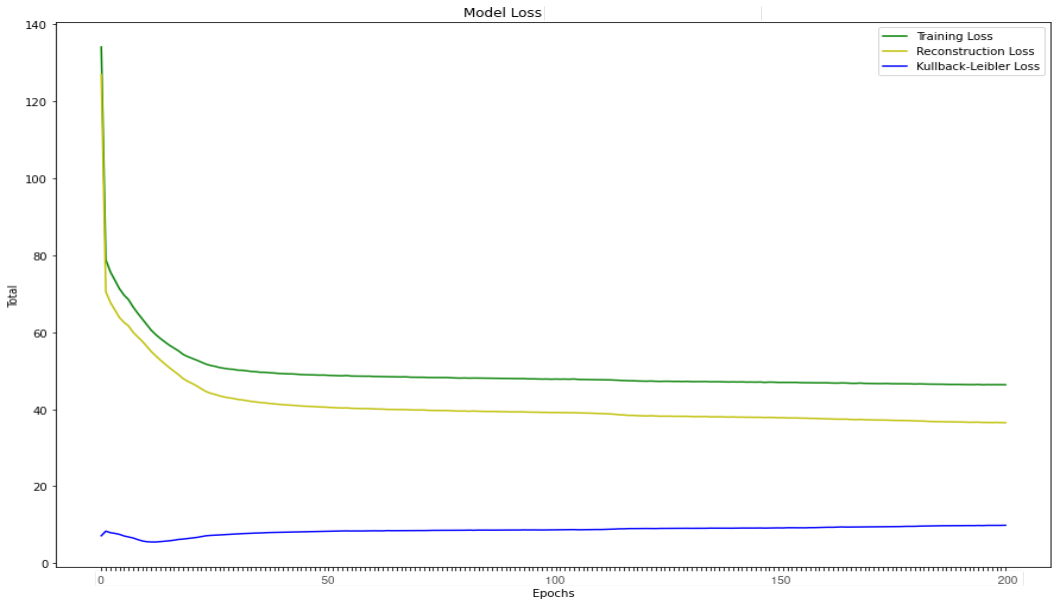


Figure 3.24: VAE Losses on experiments III

Chapter 4

Conclusions and Future Works

4.1 Conclusions

The aim of this project was to serve as a Proof of Concept on how MRI brain images could be artificially generated with Variational Autoencoders which ultimately would serve to enhance existing or new datasets to improve model accuracy by having bigger samples of data.

According to the original aim of the project, it can be concluded that, as a Proof of Concept, Variational Autoencoders can create images artificially. This project requires to be continued so that those created images could be interpreted as real by an expert eye.

Training the different models and convolutional networks with also different parameters, produced similar images. That leads to conclude that maybe the IXI dataset was too small or that the selected approach to decide which images to pre-process and use was not the best. While thesis' dataset generated (and used) 16k images, BraTS [7] uses 165k, ten times more images for anomaly detection models. So, it is clear that the number of images used in this project is far from being appropiate.

The models were able to create images that overall can be seen as brain images in terms of sizes and forms, but the brains are not detailed, instead, they look blurred. This is the main area of improvement and were a second phase of the project should focus.

Since this project implemented both Variational Autoencoders and Vector-Quantized VAEs it is ready to show the benefits of using one or the other. While they had the same imaging results due to previous conclusions, the latter has shown better learning times by reducing about 5-10 seconds on each epoch train.

The project timeline was a little short to execute some additional experiments that could have given better outcomes out of this proof of concept and it looks really short to achieve the final goal of the models. It must be taken into consideration that the researching aspects of the project were a lot, including MRI learning and cutting edge VAE models which require to learn and implement losses calculations and latent space fundamental concepts. Also, training any of the 2 models requires few hours and it would require more in case of increasing the size of the dataset.

It looks the right decision to have taken the opportunity to train VAEs for image creation rather than using Autoencoders to reconstruct existing images since VAEs were not only capable of that but

also of creating new images from the latent space. Using Autoencoders would have required less efforts in research while focusing on tuning model results from previous existing and documented projects.

Using Google Colab and Keras library proved to be the right tools to use, both of them boosted the project thanks to the community behind who have created reference projects to implement new VAE and VQ-VAE classes for models. The downside of using Colab is that free GPUs are very limited to this kind of project and even when using the PRO+ subscription, paid unites lasted for 2 days. Maybe for this kind of projects, dedicated VMs would be used to not block the project activities.

4.2 Future Work

As described in conclusions, this thesis outcomes just prove that VAEs and VQ-VAEs are learning from the dataset but the cannot create real images today. To reach that goal, several future works are foreseen.

If the project sticks to use IXI dataset and work with 2D slices, the images should be intensively processed by applying skull stripping and, at least, adding noise and centering images as pre-processing techniques that might improve the dataset quality.

Clearly, the KL-Divergence issue has to be cleared out. Maximizing ELBOs require to minimize both reconstruction loss and kl-divergence, it is not an option to let kl-divergence to increase. Another related task is to work on the formula that aggregates reconstruction loss and kl-divergence as training loss. Maybe applying factors is needed.

Talking about losses, a nice-to-have task is to be able to mark which are the values that are acceptable. For example, in case of reconstruction loss, a loss of 0 means that images are identical, but what does mean a loss of 30? or a loss of 100? If acceptable losses can be determined the expert eye would not be required to declare a model as good or bad.

In case that the IXI dataset could be extended with other MRI datasets, for instance BraTS [7], the models should be trained with them first, and then with both. Exploring results with larger datasets is key to test the implemented VAE and VQ-VAE models' quality.

Acronyms

ceVAE Context-Encoder Variational Autoencoder. 10, 11

CNN Convolutional Neural Network. 18, 20–22, 30

MRI Magnetic Resonance Image. 18, 30, 33

MSE Mean-Squared Error. 19

PCA Principal Component Analysis. 8, 9

PNG Portable Network Graphics. 18, 24

RGB Red Green Blue. 17

StRegA Segmentation Regularised Anomaly. 10, 11

VQ-VAE Vector-Quantized Variational Autoencoder. v, 10, 19, 20, 25

Glossary

IXI IXI dataset is a collection of 600 Brain MR Images, in 3-D, collected from 3 different hospitals in London. The images are scanned in 1.5 and 3 Tesla which is the range of the quality that we could usually find in hospitals around the world. 9

KDD KDD cup 1999 network intrusion dataset. 9

KPI Key Performance Indicator. 7, 9

MNIST The MNIST database of handwritten digits used for training models. 9

Bibliography

[1] Jinwon An and Sungzoon Cho. *Variational Autoencoder based Anomaly Detection using Reconstruction Probability.* Cornell University https://arxiv.org/, 2015.

[2] Matthew Brett, Chris Markiewicz, Michael Hanke, Marc-Alexandre Côté, Ben Cipollini, Paul McCarthy, and Chris Cheng. Nibabel. https://nipy.org/nibabel/.

[3] Jorge Cardoso. King's accelerates synthetic brain 3d image creation using ai models powered by cambridge-1 supercomputer. https://www.kcl.ac.uk/news/synthetic-brain-3d-image-creation-ai-models-powered-by-cambridge-1-supercomputer.

[4] Soumick Chatterjeea, Alessandro Sciarra, Max Dunnwald, Pavan Tummala, Shubham Kumar Agrawal, Aishwarya Jauhari, Aman Kalra, Steffen Oeltze-Jafra, Oliver Speckc, and Andreas Nurnberger. *StRegA: Unsupervised Anomaly Detection in Brain MRIs using a Compact Context-encoding Variational Autoencoder.* Cornell University, 2022.

[5] François Chollet. Variational autoencoder, 2021. https://keras.io/examples/generative/vae/.

[6] Bai Dash. Introduction to skull stripping (image segmentation on 3d mri images). https://www.analyticsvidhya.com/blog/2021/06/introduction-to-skull-stripping-image-segmentation-on-3d-mri-images/.

[7] Spyridon Bakas et al. Rsna-asnr-miccai brain tumor segmentation (brats) challenge 2021, 2021. http://braintumorsegmentation.org/.

[8] Seth Hettich and SD Bay. The uci kdd archive, 1999. http://kdd.ics.uci.edu.

[9] Jeremy Jordan. Variational autoencoders, 2018. https://www.jeremyjordan.me/variational-autoencoders/.

[10] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The mnist database. http://yann.lecun.com/exdb/mnist/.

[11] Sayak Paul. Vector-quantized variational autoencoders, 2020. https://keras.io/examples/generative/vq_vae/.

[12] Martin Riva. Batch normalization in convolutional neural networks, 2022. https://www.baeldung.com/cs/batch-normalization-cnn.

[13] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, Jie Chen, Zhaogang Wang, and Honglin Qiao. *Unsupervised anomaly detection via variational autoencoder for seasonal KPIs in web applications*, pages 187–196. Proceedings of the 2018 World Wide Web Conference, 2018.

[14] David Zimmerer, Fabian Isensee, Jens Petersen, Simon Kohl, and Klaus Maier-Hein. *Unsupervised anomaly localization using variational auto-encoders.* German Cancer Research Center (DKFZ), Heidelberg, Germany, 2019.

[15] David Zimmerer, Simon A.A. Kohl1, Jens Petersen1, Fabian Isensee1, and Klaus H. Maier-Hein1. *Context-encoding Variational Autoencoder for Unsupervised Anomaly Detection.* Division of Medical Image Computing, German Cancer Research Center (DKFZ), Heidelberg, Germany https://arxiv.org/, 2018.