IMA206

Project Report

2022 - 2023

Variational Auto-Encoders for cardiac shape modeling

ACDC Database

Code available here

Charles Monté, Marion Protard, Jeanne Malécot
Under the supervision of Loïc Le Folgoc *and* Elsa Angelini

I Introduction

HAVING a precise model of cardiac shape will enable us to deepen our understanding of cardiovascular pathologies and their mechanisms. The possibility of identifying specific shape patterns associated with certain pathological conditions could lead to interesting new diagnostic and therapeutic information.

As part of this project, we set out to develop a Variational Autoencoder (VAE) specifically designed for modeling **the shape of cardiac images representing the segmentation of a heart** into 4 classes: the two ventricles (left and right), which we will name LV and RV, the myocardium, which will be named MYO and the background.

The Variational Autoencoder is based on an architecture that enables complex data to be represented and generated by learning a continuous latent representation. Using this approach, we aim to capture the essential features of **segmented cardiac images/cardiac shapes** from a volumetric medical image dataset.
The main objective of this project is to provide an efficient method for **cardiac shape** modeling, exploiting the generation and compression capabilities of VAEs and the dimensional interpretability of the generated latent space. We have focused on creating a model capable of learning specific anatomical variations and deformations of the heart, while ensuring a consistent and interpretable latent representation.

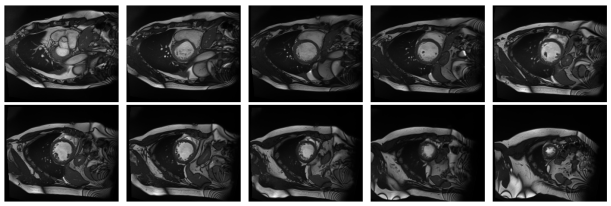

Figure 1: All MRI image slices for one patient in the ACDC dataset, during diastole

We will use the ACDC dataset[1].
Created by researchers, it has been designed to evaluate different segmentation and classification algorithms.

[1]ACDC dataset : https://www.creatis.insa-lyon.fr/Challenge/acdc/databases.html

This dataset contains heart MRI images during diastole and systole for 150 patients, often composed of around 10 slices, and their segmentation according to the regions of interest mentioned above.
It has already been split into a training set of 100 MRIs and a testing set of 50 but we decided to split it by our own means after the slice separation step.

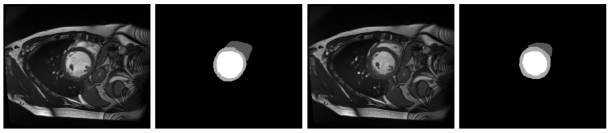

Figure 2: One slice during diastole, systole and corresponding segmentations

In this report, we will present the objectives of the project, the method we implemented, the problems we encountered and the solutions we found to address them, the results obtained and the criticisms we can make of them.

II Objectives

The first objective of this project is to create an accurate model for the shape of the heart. Our main objective has been to develop a model capable of incorporating anatomical variations as well as specific deformations of the heart based on volumetric medical image data.
By learning our variational auto-encoder (VAE), our intention has been to acquire a continuous latent representation that will be able to generate realistic cardiac shapes and consistently illustrate different anatomical configurations.

The second objective of this project is to make the cardiac shape model generated by VAE interpretable. We aim to identify significant latent features that influence cardiac shape and associate them with well-defined anatomical concepts. This will increase confidence in the model's predictions and facilitate analysis of the results.

III Method

To address this issue, we came up with the following method.

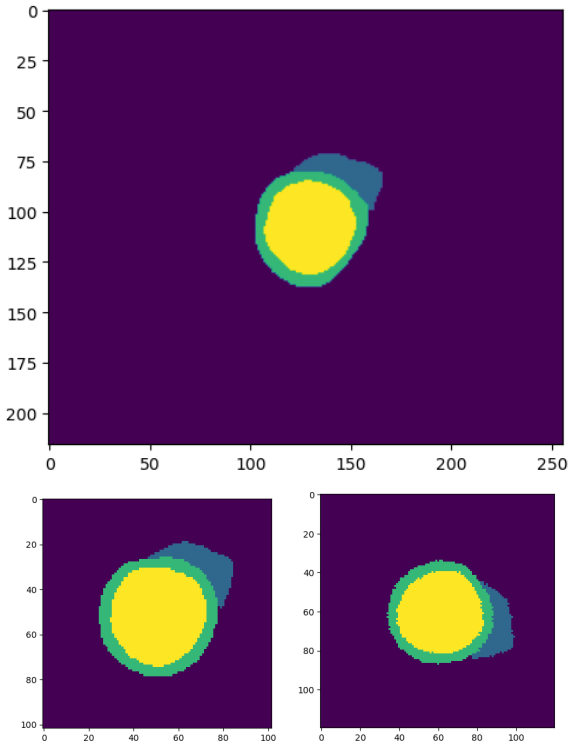

Figure 3: Original, croped and rotated slice

Data download

We imported the data: a set of 150 cardiac volume image segmentations, from which we recovered all slices containing information, i.e. if at least one pixel is part of a class, then we keep this slice. This way, we were left with 2785 different slices, which we then split by keeping 50% for the training set, 20% for the validation set, and 30% for the test set. While the training set has been used for training the VAE architecture, the validation set was used to implement early stopping while giving us a more general status on the training of the model. The test set only was used to rate the reconstruction capacities of the model.

Preprocessing

In order to remove spurious sources of variation before training our models, we implemented a number of different protocols: First, we centered the image with respect to the middle of the left cavity and tilted it so that the respective cavity centers were aligned horizontally. Next, we cropped the volume, keeping only one cube around the segmentations, allowing us to only keep the information that is important, by cropping the background, while preserving the natural size variations of the LV, which would be hindered by the next step of the preprocessing: a resizing step. All slices needed to be in an identical shape for training, which is why we chose to have a cube around the segmentation, this way we could have square slices that we could resize while keeping the natural and scientific relative sizes and shapes of the ventricles and the myocardium.

VAE Construction

We created our VAE using Pytorch. We decided to implement the architecture proposed by Nathan Painchaud et al[2] for a constrained VAE made to work on the ACDC dataset The encoder is made of four convolutional blocks, each one consisting of two convolutional layers with ELU activation, both with a kernel size of 3, the first one with a stride of 2 and the second with a stride of 1, both with same padding. The very last convolutional layer of the encoder does not have an activation. The decoder has a similar structure to the one of the encoder, it is made of four blocks preceded by a linear layer putting the size 32 vector from the latent space into the input size of the encoder, which is the same as the output size of the encoder. Its structure mirrors the encoder, with each block consisting of two layers, a transposed convolution, of kernel size 2, stride 2 and no padding, and a convolution layer of kernel size 3, stride 1 and same padding. After the decoder, a final convolution layer of kenerl size 3, stride 1 and same padding is added to enforce the output of a pixel-wise score for each class. The number of features for the first convolutional block is set at 48 and doubles at each convolutional block in the encoder, which means we end up

[2]Nathan Painchaud, Youssef Skandarani, Thierry Judge, Olivier Bernard, Alain Lalande, and Pierre-Marc Jodoin, "Cardiac Segmentation with Strong Anatomical Guarantees" IEEE TRANSACTIONS ON MEDICAL IMAGING

at 384 features by the end of the last block. All the parameters, ranging from the number and parameters of convolutional blocks, to the activation functions and the size of the latent space have been directly taken from Painchaud et al's paper. The model takes as input the image with 4 channels corresponding respectively to background (0), right ventricle (R) and myocardium (G) left ventricle (B), thus of dimension 4x256x256. The encoder outputs an image of dimension 384x16x16. This image is then projected into the latent space using two different linear layer, which will output the parameters μ and Σ of the distribution of the latent space, which will then be used to sample a random vector following that distribution before making this random vector go throught the decoder, which will output a 4 channel 256x256 image.

Loss functions definition

We have defined a loss function composed of two terms. The first term is the reconstruction error term: this is a sum of the opposite in regards to 1 of the Dice Score, named Dice Loss, that measures the dissimilarity between the input image and the image produced by the decoder, written just below, and the Cross-Entropy loss, also measuring the dissimilarity between the input image and the output of the decoder.

$$1 - \frac{2|X \cap Y}{|X| + |Y|} = \sum_k \frac{2\sum_i x_{ik} y_{ik}}{\sum_i x_{ik} + y_{ik}}$$

The second term is the Kullback-Leibler divergence. This is a measure of dissimilarity between two probability distributions, which in our case compares the Gaussian distribution with the parameters μ and Σ obtained after encoding, and a centered reduced Gaussian distribution.

$$KL(P||Q) = \sum_{x \in \mathbb{X}} P(x) * log\left(\frac{Q(x)}{P(x)}\right)$$

As explained in part IV, throughout the training, the loss caused us a lot of problems, which led us into adding a β term in front of the KLD part of the loss, allowing us to enforce even more the link between the latent space's distribution and a centered reduced Gaussian distribution. This modification was not made by Painchaud et al.

In the end, the loss used has been this one :

$$Loss = Dice(y_p, y) + CE(y_p, y) + \beta KL(P||Q)$$

where $Dice(y_p, y)$ and $CE(y_p, y)$ respectively represent the Dice and the Cross-Entropy Loss between the output y_p and the input y of the VAE, and where $KL(P||Q)$ is the Kullback-Leibler divergence between the obtained distribution with parameters μ and Σ, and a centered reduced Gaussian distribution

Model training

We then trained our model on mini-batches of slices size 100 for nearly 180 epochs, using Adam optimizer with a learning rate for 6×10^{-5} and a L2 weight penalty with $\lambda = 10^{-2}$. All those parameters were chosen by taking the exact parameters that were used by Painchaud et al for their training.

IV Problems encountered

During the training, a small but very disabling range of problems got in our way, those problems notably appeared through the choice of loss, but also during the modeling and exploration of the latent space.

As explained above, the loss of a VAE is made up of two parts: a loss directly focused on image reconstruction, i.e. a classic model training loss, named the reconstruction loss, measuring the difference between the target and the prediction (the target here being the input image of the model and the prediction the output image of the model), but also an additional term in the loss which forces the latent space to be modeled according to a centered reduced normal distribution.

As already explained, it was quickly decided to work with a reconstruction loss that is an equal mix between a cross-entropy loss and a dice loss (which is equal to the opposite in relation to 1 of the dice score between the target and the prediction). This choice is essential be-

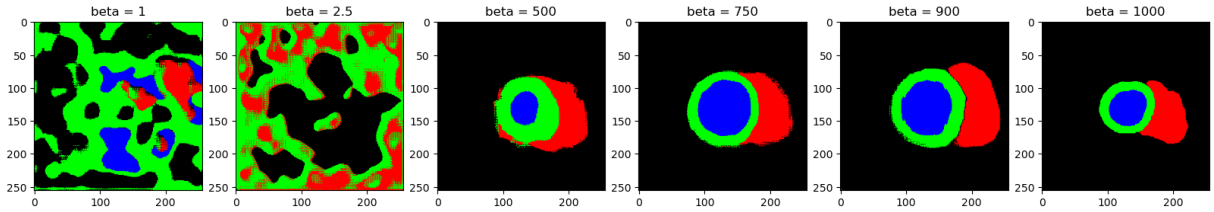

Figure 4: Evolution of generation quality in function of β

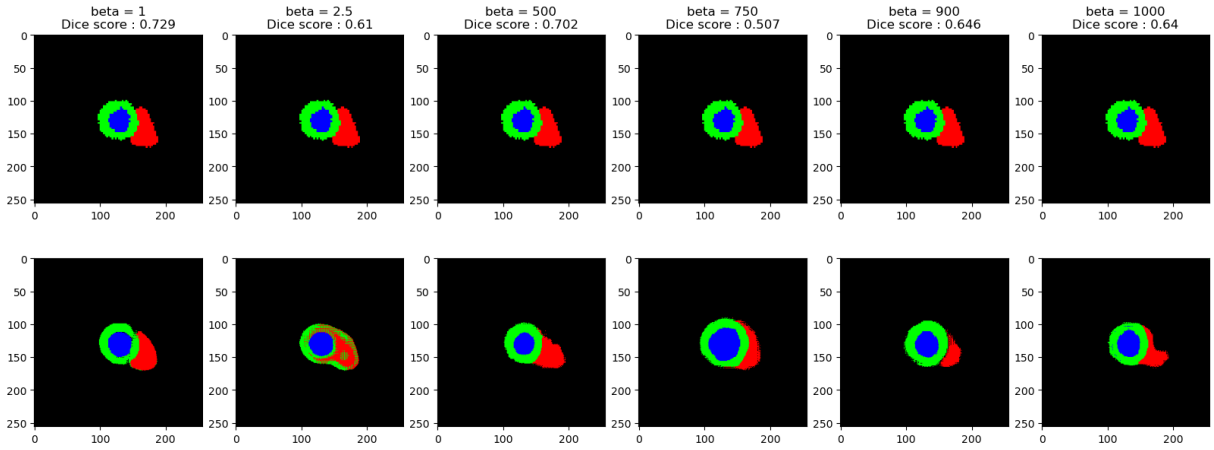

Figure 5: Evolution of reconstruction quality in function of β

cause giving too much importance to the cross-entropy would then have given too much importance to the background, which dominates in terms of the number of pixels in the slices. For the loss that ensures good modeling of the latent space, the natural choice, as during the practical works and as explained in the course, was to take the Kullback-Leibler divergence calculated between the normal distribution adapted to the representation of the latent space during training and a reduced centered Gaussian distribution. Probably due to problems in reducing these losses, and poor choices between taking the sum and the mean over the whole batch, we had a lot of trouble getting good results without modifying the distribution of the two losses in the final training loss.

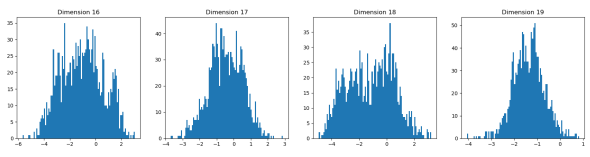

Figure 6: Some dimension value histograms for $\beta = 1$

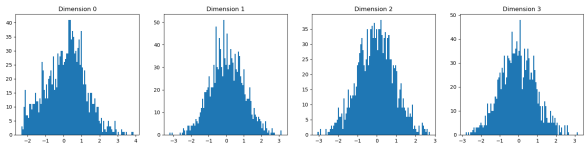

Figure 7: Some dimension value histograms for $\beta = 1000$

Our first results showed nearly identical results in the generation phase while having poor reconstruction capacities, which made us understand that the KLD had too much importance in the loss: the good latent space's representation was too important thus leading to an absence of diversity, the model did not learn a lot from the diversity inside of the training set because the reconstruction loss was not important enough. This lead to all the generated vectors in the latent space following a reduced centered Gaussian distribution to give a nearly identical output.

By trying to compensate for this problem, we over-aimed and ended up in the opposite situation, where the latent space was not distributed following a reduced centered Gaussian distribution while the reconstruction capacities of our model were really good.

The solution we ended up using was to add a β parameter in front of the KLD part of the loss, to better be able to understand how much importance we would put in each of the capacities of our model. The choice was simple, a higher β would lead to worse reconstruction capacities but a more understandable and usable latent space, while a lower β would lead to the opposite. As the objective of this project was to explore the latent space and try to find interesting dimensions, a higher β was the way we decided to go.

To our surprise, we ended up having to use a β nearly equal to 1000, i.e. giving 1000 times more importance to the KLD than to the reconstruction loss to finally have some good results. Figures 4 and 5 show the problems that each exaggeration of the β parameter can bring, as explained before, it is visible that the generation quality increases while the reconstruction quality decreases.

The problem caused by a β value that is too low is best analyzed when looking at the latent space distribution, rather than looking at the reconstructions and the generations. As we are looking for a centered reduced Gaussian distribution, we in fact want each dimension to follow a 1D centered reduced Gaussian distribution, a condition that can be easily analyzed by looking at the distributions of the values of those dimensions for each image inside of the train set.

With this type of data, the difference is directly visible. On the one hand, for $\beta = 1$ (fig. 6), at best the distributions can be approximated by a Gaussian but with the wrong parameters while a majority of the time, they can be approximated by a mixture of Gaussians. On the other hand, for $\beta = 1000$ (fig. 7), even if some distributions cannot be perfectly approximated by centered reduced Gaussian distributions (Dimension 0 for instance), they are all centered and for the majority comprised between -3 and 3.

However, the difference when trying to look at it with an approach of dimensionality reduction is a lot less visible, the results are even a bit surprising. Indeed, as we can see on figure 8, the UMAP representation of the latent space with a higher β value is not as clustered as the one with a really low β value, which is something we found surprising and did not really manage to explain outside of the fact that the latent space has less variation, which might lead the UMAP fitting into thinking that all the data is nearly the same.

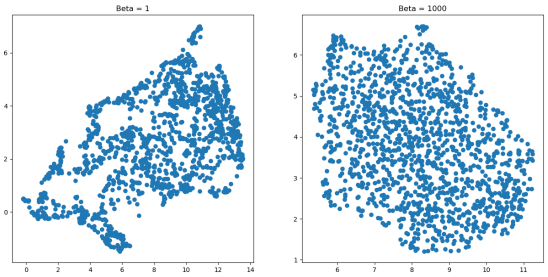

Figure 8: UMAP representations of the latent space for two values of β

V Results

The last model selected showed satisfactory results in both reconstruction and generation. During training, we noted that early stopping prevented loss function from increasing (fig. 9), i.e. the model did not overfit.

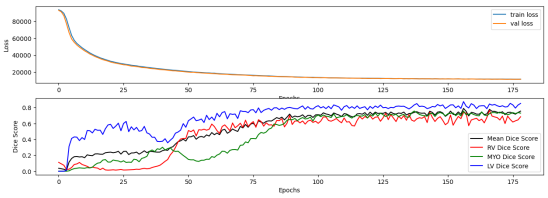

Figure 9: Loss function and dice scores evolution

We also noticed, by visualizing the evolution of the dice scores for each region, that the model seemed to learn in phases. Indeed, their evolution is not monotonous like that of the loss function.
First, it learns to reconstruct the left ventricle, then the myocardium and finally the right ventricle. The start of a new phase causes the score of the two regions not involved to fall, before causing them to rise again.

At the end of training, the average score is quite high, although the model still has more difficulty in reconstructing the right ventricle and, to a lesser extent, the myocardium. This is understandable, given their more complex

shape than the left ventricle - a large, rather regular, centered circle.

Reconstruction

Once the model had been trained, we obtained a suitable dice score of 0.75. The biggest differences between the reconstructions and their originals can be seen in the right ventricle. The results were always close in terms of scale and position, but there were some difficulties with the shape. Disappointing reconstructions have also been observed, with myocardial thickness clearly different from the original image (fig. 10).

We also wanted to test our model on images with levels of variance that had not been learned from the model; for example, with images where the two ventricles are not aligned. The result is, unsurprisingly, not as good, but still gives a dice score of 0.69 (fig. 11). This rather high figure is understandable, given that a rotation really affect the reconstruction of the right ventricle only, all the more so when the rotation has taken it to the left of the image.

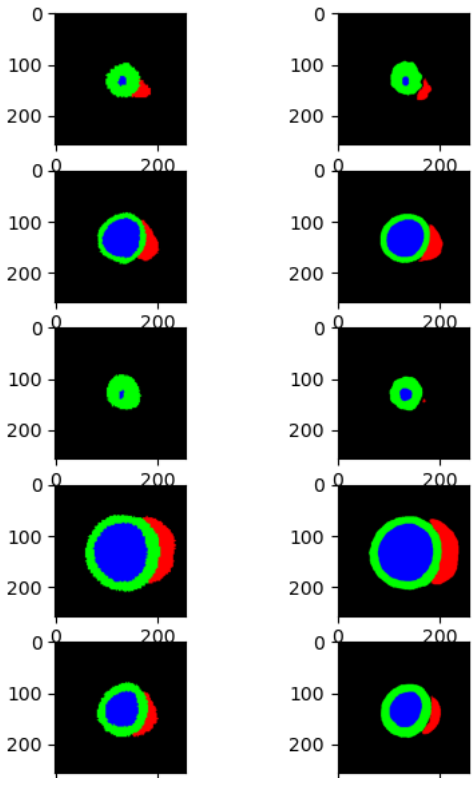

Figure 10: Reconstructed slices with optimal model

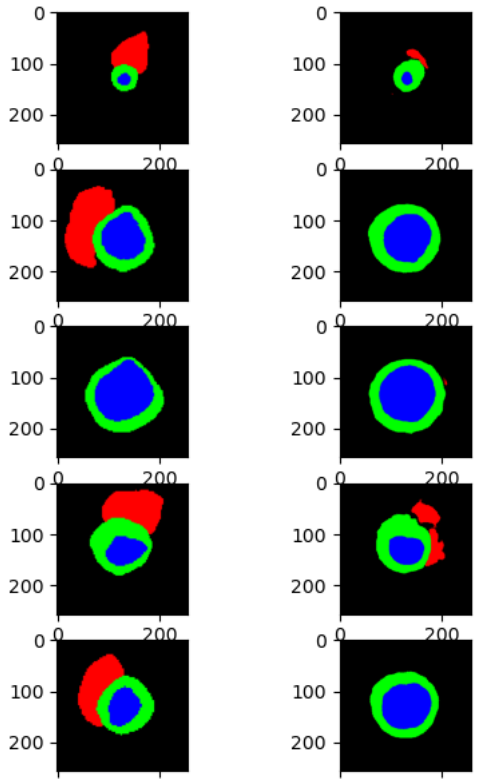

Figure 11: Reconstruction of not rotated slices with optimal model

Generation

With this fairly high beta choice, we have achieved very satisfactory generations (fig. 12). The result is realistic and rather varied segmentations.

This led us into exploring the latent space in its pure form, dimension by dimension, to try and find the dimensions that had the most impact on the generations and modified them the most. To do so, we chose the image generated by the decoder when using the vector equal to zero in the latent space. Then, we decided to generate, starting from this vector, and exploring in a linear way, images representing the variations of each dimensions. For instance, we generate a matrix full of zeros of size (50,32), then, we modified the first line so that it would represent a list of 50 samples of a linear variation ranging from -3 to 3. This way, we could see the physical actions each dimension had on the generations, we could quantify it and then choose the real dimensions of interest.

The results turned out really great, allowing

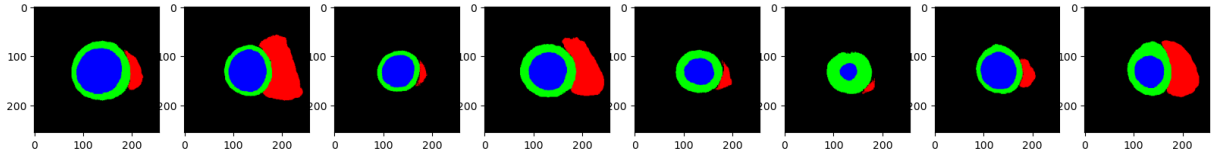

Figure 12: Generated slices with optimal model

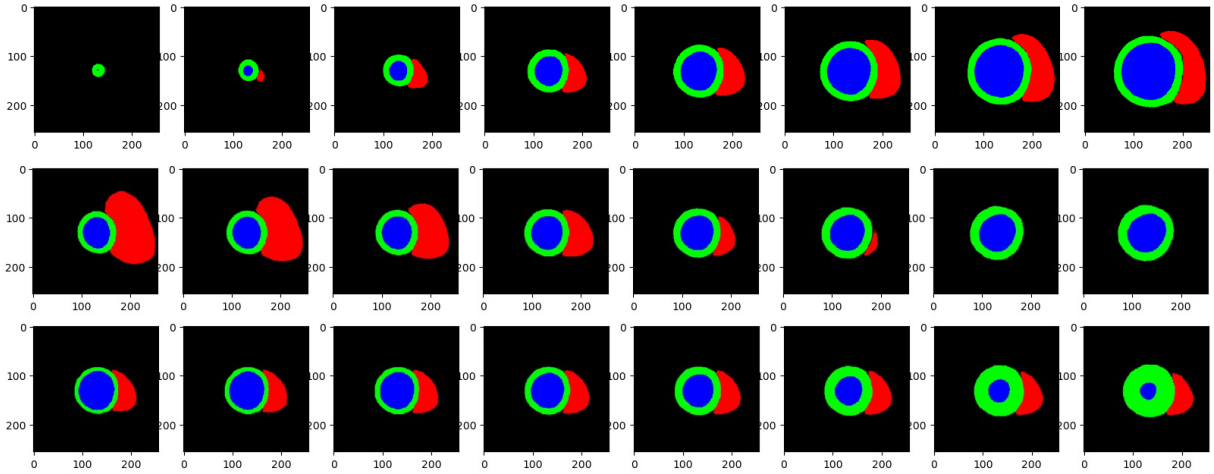

Figure 13: Examples of linear variations of dimensions 0, 26 and 15 in that order

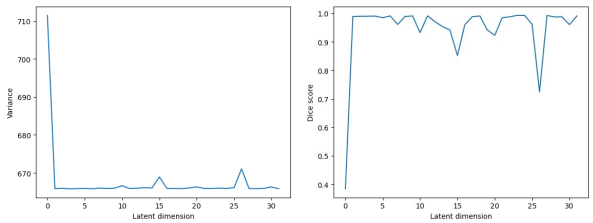

Figure 14: Values of average dice score and variance in function of the dimension modified

us to choose a lot of different parameters to modify, from which we could list: LV + MYO size, RV shape, MYO thickness, LV centering relative to MYO center on the vertical axis, LV + MYO size on the horizontal axis, LV centering relative to MYO center on the horizontal axis, LV + MYO size on the vertical axis.

As it is seen on figure 14, both the metrics show 3 dimensions as the most influent, dimension 0, dimension 26 and dimension 15 in the order of most to less influent (a higher variance or a lower average dice score over the batch both mean more influence on the generations). Those dimensions, as shown on figure 13, modify the generations while having a real scientific meaning, as dimension 0 regulates the size of the MYO+LV pair, dimension 26 the shape of the RV and dimension 15 the thickness of the MYO.

This characteristic of the latent space is a very good thing, as it could be used for real data augmentation purposes, for instance, if we lack patients with a thick MYO for the training of a segmentation model, we could generate more data like it, supposing we also have created a VAE capable of generating heart MRIs from heart segmentations. What the model lacked to succeed in is the capacity to easily modify the shape and size of the RV through a dimension of the latent space, as it looks like dimension 26 is a little too harsh and not that capable of explore all types of RV shapes.

Furthermore, what is possible to take from this exploration of each dimension is the fact that it is difficult to find real differences between different types of segmentations without a human and medical eye capable of making the difference, as the dice scores for instance, except for the 3 dimensions talked about above, rarely change. This mainly explains the difficulty UMAP had to cluster the different types of segmentations, because the classes are difficult to differentiate, as a thick MYO can be accompanied by all types of RV for example,

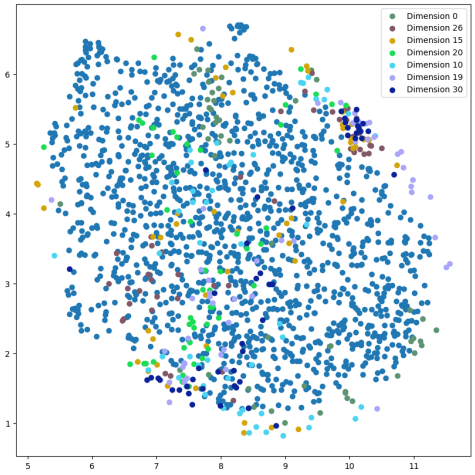

Figure 15: Positions of images obtained by modifying only one dimension in the latent space in the UMAP space

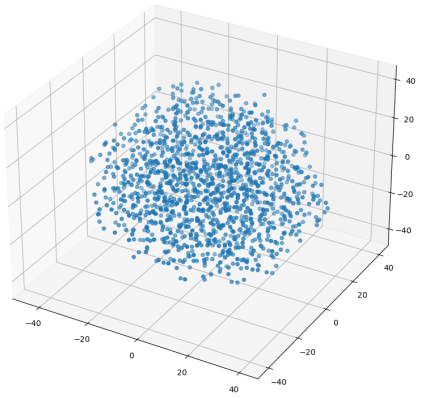

Figure 16: Representation of the latent space with dimensionality reduced with TSNE using 3 components

which explains the appearance of the UMAP we talked about in part IV. To carry even further this observation, we can look where really different segmentations and classes are placed inside of the UMAP generated before, which is what figure 15 shows us.

From that figure, it is understandable why the UMAP space had this shape, as even images with great difference between them, for instance those extracted from dimensions 26 and 15, have also a lot of points in common, which makes them be close inside of the UMAP space. This kind of conclusion is the same when trying with other dimensionality reduction algorithms like TSNE, even when used with a higher number of final components, as shown on figure 16.

VI Conclusion

Although we chose to present the model with a β value of 1000 as the optimal model, previous reconstruction results with lower values were not without interest. The results in generation were unusable, but having very precise reconstructions can be used, for example, to repair bad segmentations. Indeed, it is more interesting in practice to have a model that provides only very precise and not biologically erroneous results.

To obtain a model that performs better in both respects, we could, for example, have explored the possibility of a modulable beta: very small at first, to force the model to learn well, then larger until good reconstructions are obtained. We've only had time to try a linear variation of beta, which didn't give good results. A more relevant idea would have been to make it evolve exponentially, so that it increases significantly early on (after around 20 epochs, for example).