PRIM - AI Option

Diffusion Models for Cardiac Images

Students:

Louis LACROIX
Benjamin TERNOT

Supervisors:

Elsa ANGELINI
Loïc LE FOLGOC

2024-2025 Semester 1

Contents

1 Introduction

Learning models of anatomy has many uses in medical imaging: generating synthetic data for data augmentation in downstream models, describing normal and pathological variability in a population, visualizing the effect of a clinical or demographic factor, providing reduced and explainable feature spaces for diagnosis, helping obtain anatomically plausible organ segmentations, etc... Nowadays, we can rely more on more on deep learning based generative frameworks for this task. In particular diffusion models are the current state-of-the-art in terms of realism of generated images in computer vision and medical imaging domains.

The first objective of our project is to get familiar with diffusion models as a whole, especially in the context of medical images and cardiac shapes. For this, one important part of the project is to read theoretical articles on stable diffusion in the context of images and potentially text, and to study the mathematics and the rationale behind Diffusion Models. Then, the next important part of our project is the concrete implementation of Diffusion models. Here we want to use Diffusion Models [1] for the generation of binarized images (multiclass masks) of cardiac shapes. That is, making, implementing, training and testing a Diffusion model on a specific Data Set of cardiac shapes and masks ourselves (with required pre-processing of the DataSet). Then, one important goal is to obtain an efficient and viable model, potentially implementing the blurring diffusion models of [2] and combining blurred diffusion models with the VAE framework for improved accuracy.

The second objective of our project is a more advanced use of Diffusion Models for medical imaging. We would like to deepen our understand of Diffusion and use them for more diverse or less common tasks. Here, we especially want to add a conditional input to the Diffusion model that will condition the generation instead of pure random generation. One idea is to condition on specific heart pathologies (Simple labels). The more complex one is to use a Diffusion model directly for the segmentation of cardiac shape, therefore adapting the DDPM to the task by conditoning the generation on a specific heart shape (2D or 3D image) and training adequatly.

2 Diffusion Models

2.1 Quick history of Diffusion Models

Generative models have become a cornerstone of machine learning, with various approaches demonstrating remarkable capabilities in creating high-quality data samples in multiple data domains. Among these, diffusion models are a relatively recent addition in the last few years with unique strengths and characteristics, especially for the domain of images (The most knowable algorithm being stable diffusion for text to image generation).

The foundations of diffusion models can be traced back to the intersection of statistical physics and probabilistic machine learning, particularly non-equilibrium thermodynamics and Markov processes. The idea of progressively perturbing data to learn its structure is inspired by concepts such as the forward and reverse diffusion processes in stochastic systems.

The first significant theoretical groundwork was laid by Sohl-Dickstein et al. in their 2015 paper, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics". This work introduced the idea of iteratively corrupting data through a diffusion process and then training a model to reverse the noise. However, the computational complexity and relatively modest outcomes limited the immediate impact of this method.

In the following years, several advances built upon this foundation:
- Score-based Generative Models: Song and Ermon (2020) proposed the "Score Matching with Langevin Dynamics" framework. This approach introduced a denoising score-matching technique to estimate the gradient of the data distribution and used Langevin dynamics for sampling. This improved the mathematical clarity and performance of diffusion models.
- Denoising Diffusion Probabilistic Models (DDPMs): Ho et al. (2020) presented the now-celebrated "Denoising

Diffusion Probabilistic Models" paper, formalizing the reverse diffusion process using variational inference. DDPMs refined the training and sampling methods, making diffusion models viable for high-quality image synthesis.
- Improved Sampling Techniques: Subsequent papers, such as Song et al.'s "Denoising Diffusion Implicit Models" (DDIMs, 2021), introduced deterministic sampling techniques, dramatically accelerating the sampling process without sacrificing sample quality.
- Unified Perspective: In their 2021 paper "Score-Based Generative Modeling through Stochastic Differential Equations", Song et al. framed diffusion models within the broader mathematical context of stochastic differential equations (SDEs). This unified framework enabled new sampling strategies and theoretical insights. =
- Architectural Innovations: Recent works have incorporated cutting-edge neural architectures into diffusion models, such as U-Net-based designs to minimize the loss function, which allowed to learn large number of parameters for big data sets, as well as generating images from text using pre-trained image and text models, and which are commonly used in state-of-the-art implementations like DALL-E 2, Stable Diffusion, and Imagen. These architectures provide the necessary capacity to handle high-dimensional data, such as images or videos, with impressive fidelity.
- Extensions and Applications: Diffusion models have been extended beyond image synthesis to domains like audio generation, molecular design, and reinforcement learning. Papers such as "Latent Diffusion Models" (Rombach et al., 2022) introduced techniques to model latent spaces, significantly reducing computational requirements by diffusing into smaller spaces while maintaining high quality through time-step encoding.

2.2 Theory behind Denoising Diffusion Probabilistic Models (DDPM=

Diffusion models are only one kind of generative model. There also exists Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Flow-based models. While these have achieved notable success in generating high-quality samples, each comes with inherent limitations:

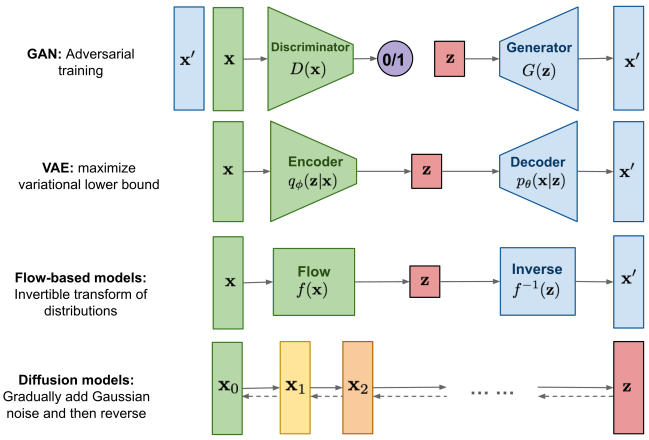

Figure 1: Overview of different generative models [5]

- GANs are lauded for their high-quality outputs but suffer from training instability and a tendency for mode collapse, which reduces the diversity of generated samples.
- VAEs rely on a surrogate loss function, which can sometimes lead to a trade-off between reconstruction accuracy and sample quality.
- Flow-based models require carefully designed architectures to ensure reversible transformations, adding complexity to their implementation.

Diffusion models take a fundamentally different approach, rooted in Markov processes and stochastic modeling. They define a sequence of diffusion steps to incrementally add random noise to data during training. The model then learns to reverse this process, transforming noise back into meaningful data samples. Unlike VAEs or flow models, diffusion models use a fixed training procedure, where the latent space has the same dimensionality as the original data (there also exists latent diffusion where the diffusion happens in a smaller latent space, it drastically

improves training time)

Key advantages of diffusion models include:
- Stability during training, as the noise addition and reversal are well-defined processes.
- High sample diversity, avoiding mode collapse seen in GANs.
- Theoretical elegance, connecting deeply with probabilistic principles and thermodynamic analogies.
- Their ability to learn the whole distribution of complex data, from high level abstract features to low level, highly detailed features
- Their ability to show a diffusion process to the user which has a particular aesthetic visual effect
- With the advancements in theory, architecture, and sampling strategies, diffusion models have emerged as one of the most powerful tools in modern generative modeling, achieving state-of-the-art results in multiple domains

2.2.1 Forward diffusion process

Given a data point sampled from a real data distribution $\mathbf{x}_0 \sim q(\mathbf{x})$, let us define a forward diffusion process in which we add small amount of Gaussian noise to the sample in T steps, producing a sequence of noisy samples $\mathbf{x}_1, \ldots, \mathbf{x}_T$. The step sizes are controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^T$.

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

The data sample gradually loses its distinguishable features as the step becomes larger. Eventually when $T \to \infty$, $\mathbf{x}_T$ is equivalent to an isotropic Gaussian distribution.

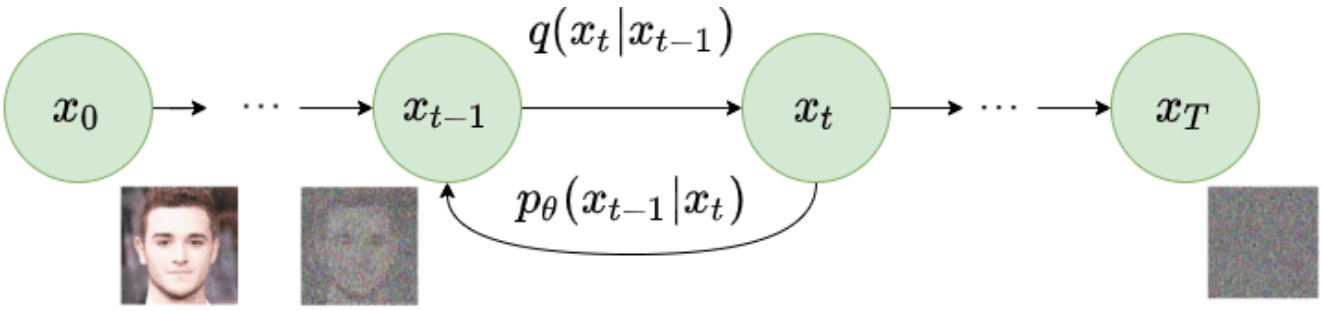

Figure 2: Forward Diffusion Process [5]

Using a simple re-parametrization trick, we can show that $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$ with $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. Therefore, we have a much simpler expression of the probability of image at time t depending directly on $\mathbf{x}_0$, which is better for training and learning the reverse diffusion process.

2.2.2 Reverse diffusion process, training loss and parameters

If we can reverse the above process and sample from $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, we will be able to recreate the true sample from a Gaussian noise input, $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Note that if β_t is small enough, $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ will also be Gaussian. Unfortunately, we cannot easily estimate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ because it needs to use the entire dataset and therefore we need to learn a model p_θ to approximate these conditional probabilities in order to run the reverse diffusion process.

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

To learn the reverse diffusion process, we need to learn a model (most of the time a neural network, especially U-Net for images) to approximate the conditioned probability distributions in the reverse diffusion process, $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$.
We would like to train $\boldsymbol{\mu}_\theta$ to predict $\tilde{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right)$, (We get this equation from a few tricks, cf [5]).

Because $\mathbf{x}_t$ is available as input at training time, we can reparameterize the Gaussian noise term instead to make it predict $\boldsymbol{\epsilon}_t$ from the input $\mathbf{x}_t$ at time step t:

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\Big)$$
$$\text{Thus } \mathbf{x}_{t-1} = \mathcal{N}(\mathbf{x}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\Big), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

The loss term is parameterized to minimize the difference from $\tilde{\boldsymbol{\mu}}$:

$$\begin{aligned}
L_t &= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[\frac{1}{2\|\boldsymbol{\Sigma}_\theta(\mathbf{x}_t,t)\|_2^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t,t)\|^2\Big] \\
&= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[\frac{1}{2\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\Big) - \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\Big)\|^2\Big] \\
&= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\|^2\Big] \\
&= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t,t)\|^2\Big]
\end{aligned}$$

Empirically, Ho et al. (2020) found that training the diffusion model works better with a simplified objective that ignores the weighting term:

$$\begin{aligned}
L_t^{\text{simple}} &= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\Big[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\|^2\Big] \\
&= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\Big[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t,t)\|^2\Big]
\end{aligned}$$

The final simple objective is: $L_{\text{simple}} = L_t^{\text{simple}} + C$ where C is a constant not depending on θ .

Some improvements from the original DDPM article where made to improve log likelihood ratios on the data:
- Firstly the variance β_T schedule during the forward diffusion process, was found empirically to have better results when $\beta_t = \text{clip}(1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}, 0.999) \quad \bar{\alpha}_t = \frac{f(t)}{f(0)} \quad$ where $f(t) = \cos\left(\frac{t/T+s}{1+s}\cdot\frac{\pi}{2}\right)^2$
- To learn $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)$, they first started to fix the term (no learning), then they changed the simple objective (than doesn't have this term) to $L_{\text{hybrid}} = L_{\text{simple}} + \lambda L_{\text{VLB}}$ and minimized $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t,t) = \exp(\mathbf{v}\log\beta_t + (1-\mathbf{v})\log\tilde{\beta}_t)$
Then, the final algorithm for vanilla diffusion looks like this:

Algorithm 1 Training

1: **repeat**
2: $\quad\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $\quad t \sim \text{Uniform}(\{1,\dots,T\})$
4: $\quad\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
5: $\quad$Take gradient descent step on
$$\nabla_\theta\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
2: **for** $t = T,\dots,1$ **do**
3: $\quad\mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\right) + \sigma_t\mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Figure 3: Algorithm Summary [5]

2.3 Conditional generation, latent diffusion, stable diffusion and improved versions

For now, DDPM was only able to sample to generate random samples from the learned distribution via a diffusion process. The real power of diffusion models comes from the ability to condition the diffusion process on multiple data (To which a correlation has to be learned during training) to generate specific data from prompts. The prompts can be any kind of data, from classes, to images, texts, sounds, etc...

2.3.1 Conditional generation

To explicit incorporate class information into the diffusion process, Dhariwal ̆0026 Nichol (2021) trained a classifier $f_\phi(y|\mathbf{x}_t, t)$ on noisy image $\mathbf{x}_t$ and use gradients $\nabla_{\mathbf{x}} \log f_\phi(y|\mathbf{x}_t)$ to guide the diffusion sampling process toward the conditioning information y (e.g. a target class label) by altering the noise prediction. Recall that $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ and we can write the score function for the joint distribution $q(\mathbf{x}_t, y)$ as following

$$\begin{aligned}\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t, y) &= \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log q(y|\mathbf{x}_t) \\ &\approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) + \nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t) \\ &= -\frac{1}{\sqrt{1-\bar{\alpha}_t}}(\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) - \sqrt{1-\bar{\alpha}_t}\nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t))\end{aligned}$$

Thus, a new classifier-guided predictor would take the form as following : $\bar{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t) = \boldsymbol{\epsilon}_\theta(x_t, t) - \sqrt{1-\bar{\alpha}_t}\nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t)$
To control the strength of the classifier guidance, we can add a weight w to the delta part, $\bar{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t) = \boldsymbol{\epsilon}_\theta(x_t, t) - \sqrt{1-\bar{\alpha}_t}\ w\nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t)$
Such models are able to achieve better results than SOTA generative models like BigGan and with some modifications on the U-Net architecture have better results than GAN (Larger model depth/width, more attention head, multi-resolution attention, rescaling and adaptive group norm). (cf. Dhariwal Nichol (2021)).
There exists other techniques for conditioning like Classifier-Free Guidance (Cf. [**AISummerclassifier-free-guidance2024**]) where you don't need an independant classifier f_ϕ but where you train two generative models, one unconditioned and the other conditioned by alternatively cutting the class condition to be able to generate "pure" samples.

FID and IS score The FID is inspired by the earlier inception score (IS) metric which evaluates only the distribution of generated images. The FID metric does not replace the IS metric; classifiers that achieve the best (lowest) FID score tend to have greater sample variety while classifiers achieving the best (highest) IS score tend to have better quality within individual images. These metrics are the state of the art for assessing the generation of images, in GAN as well as in Diffusion Models. We generally seek a form a tradeoff beetween the two metrics, depending on the use we want for our model.

Formula for FID score:
$d_F(\mu, \nu) := \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathbb{R}^n \times \mathbb{R}^n} \|x-y\|^2 \, \mathrm{d}\gamma(x,y)\right)^{1/2}$ where $\Gamma(\mu, \nu)$ is the set of all measures on $R^n \times R^n$ with marginals μ and ν on the first and second factors respectively.

2.3.2 Latent diffusion, text to image models, and diffusion models for segmentation

Latent diffusion Latent diffusion models are models that diffuse in a latent space of the image space. The latent space is learned using an auto-encoder. The whole diffusion process, forward and backward happens in the latent space. The diffusion and denoising processes happen on the latent vector f_z.

Architecture and Text to image The denoising model is a time-conditioned U-Net. The U-net incorporates a cross-attention mechanism to handle flexible conditioning information for image generation (It can be class labels, semantic maps, blurred variants of an image). The design is equivalent to fuse representation of different modality

into the model with a cross-attention mechanism. Each type of conditioning information is paired with a domain-specific encoder τ_θ, to project the conditioning input y to an intermediate representation that can be mapped into cross-attention component $\tau_\theta(y) \in \mathbb{R}^{M \times d_\tau}$:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\Big(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\Big) \cdot \mathbf{V}$$

$$\text{where } \mathbf{Q} = \mathbf{W}_Q^{(i)} \cdot \varphi_i(\mathbf{z}_i),\ \mathbf{K} = \mathbf{W}_K^{(i)} \cdot \tau_\theta(y),\ \mathbf{V} = \mathbf{W}_V^{(i)} \cdot \tau_\theta(y)$$

$$\text{and } \mathbf{W}_Q^{(i)} \in \mathbb{R}^{d \times d_\epsilon^i},\ \mathbf{W}_K^{(i)}, \mathbf{W}_V^{(i)} \in \mathbb{R}^{d \times d_\tau},\ \varphi_i(\mathbf{z}_i) \in \mathbb{R}^{N \times d_\epsilon^i},\ \tau_\theta(y) \in \mathbb{R}^{M \times d_\tau}$$

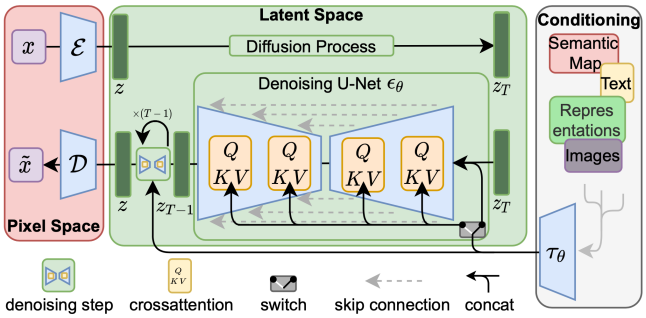

Figure 4: Latent Diffusion, Rombach & Blattmann, et al. 2022

Here, each domain-specific encoder τ_θ can be taken from an already trained big model, like the CLIP text encoder for example.

Model architecture The U-Net architecture is the main model for Diffusion, that can learn image to image functions. They have different parts:

1) Downsampling: Each step consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a ReLU and a 2x2 max pooling with stride 2. At each downsampling step, the number of feature channels is doubled.
2) Upsampling: Each step consists of an upsampling of the feature map followed by a 2x2 convolution and each halves the number of feature channels.
3) Shortcuts: Shortcut connections result in a concatenation with the corresponding layers of the downsampling stack and provide the essential high-resolution features to the upsampling process.

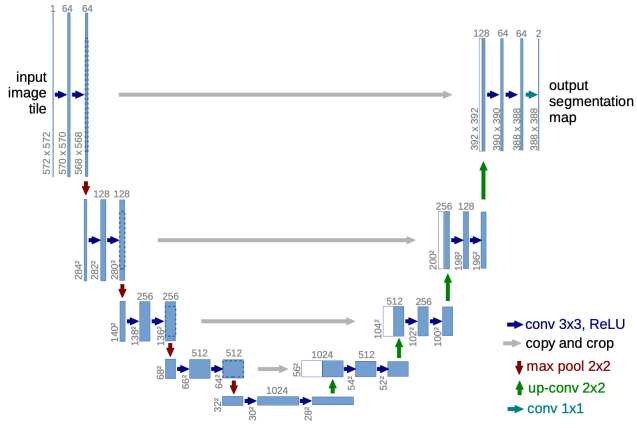

Figure 5: U-Net Architecture, Ronneberger, 2015

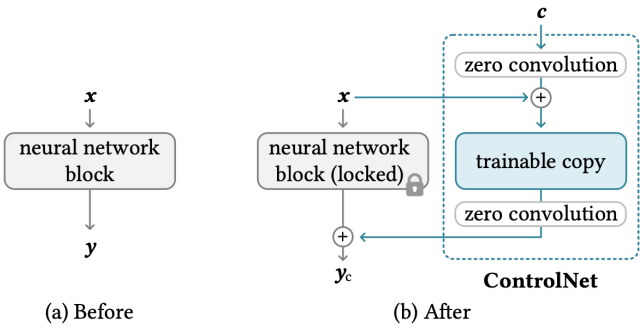

Figure 6: ControlNet Architecture [6]

ControlNet ControlNet [6] introduces conditioning the generation of the model on specific images. It can be used to "fill" lower dimensional images like Canny edges, Hough lines, user scribbles, human post skeletons, segmentation maps, depths and normals. It can also be used for segmentation, going from higher dimensional space to a lower one.

Given a neural network block $\mathcal{F}_\theta(.)$, ControlNet does the following:

- First, freeze the original parameters θ of the original block.
- Then, clone it to be a copy with trainable parameters θ_c and an additional conditioning vector .
- Use two zero convolution layers, denoted as and $\mathcal{Z}_{\theta_{z1}}(.;.)$ and $\mathcal{Z}_{\theta_{z2}}(.;.)$, which is 1x1 convo layers with both weights and biases initialized to be zeros, to connect these two blocks. Zero convolutions protect this back-bone by eliminating random noise as gradients in the initial training steps.
- The final output is: $\mathbf{y}_c = \mathcal{F}_\theta(\mathbf{x}) + \mathcal{Z}_{\theta_{z2}}(\mathcal{F}_{\theta_c}(\mathbf{x} + \mathcal{Z}_{\theta_{z1}}(\mathbf{c})))$

Variants of diffusion models Some improvements or variants of diffusion models can be noted. De-noising diffusion implicit model (DDIM) speed up the training while attaining better quality images by following only a subset of steps from the forward diffusion during training. Consistency models learn direct mapping between x_t and x_0. Blurring diffusion models [2] use a different noise function in the Fourier space instead of Gaussian noise, which allows for different kind of properties and outputs. Score-based models that learn directly the gradient $\nabla_x log(q(x))$ of the distribution. This then allows us to connect Diffusion Models to Stochastic Differential equations, Langevin dynamics and Information Theory. (cf. [4])

3 ACDC data set

3.1 Overview

The overall ACDC data set was created from real clinical exams acquired at the University Hospital of Dijon. It is a commonly used data set for training models in the medical field, since it represents a data set of RMI-labeled cardiac images with their segmentation. Thus, it can be used for both classification and segmentation.
The set is divided into training and testing subsets, containing respectably data from 100 and 50 patients.
For each patient, we have 5 images at our disposal:

- a 4D RMI of the heart, that is, 3D image along a temporal axis to have the image during a whole cycle of a heartbeat
- 2 3D-RMI images corresponding to the peak of diastole (ED) and systole (ES)
- 2 3D-segmentation (ground truth) images of both ED and ES RMIs, in which the right ventricle endocardium (RV), left ventricle endocardium (LV), left ventricle myocardium (MYO), and background are segmented

In addition, some metadata are given for each patient such as:

- the frames corresponding to the ED and ES in the 4D image
- the medical condition of the patient or group (normal - NOR, previous myocardial infarction - MINF, abnormal right ventricle - RV, dilated cardiomyopathy - DCM, hypertrophic cardiomyopathy - HCM)
- height and weight of the patient
- nbFrame, the number of frame of the 4D image

The 150 patients were selected to ensure 30 patients for each cardiac condition, distributed with 20 in the training subset and 10 in the testing subset. Figure 7 shows a 2D slice of the available images for 2 patients, along with their metadata. Here, the images we will use are the segmentation mask images, thus we have 2 per patient, one during diastole and one during systole.

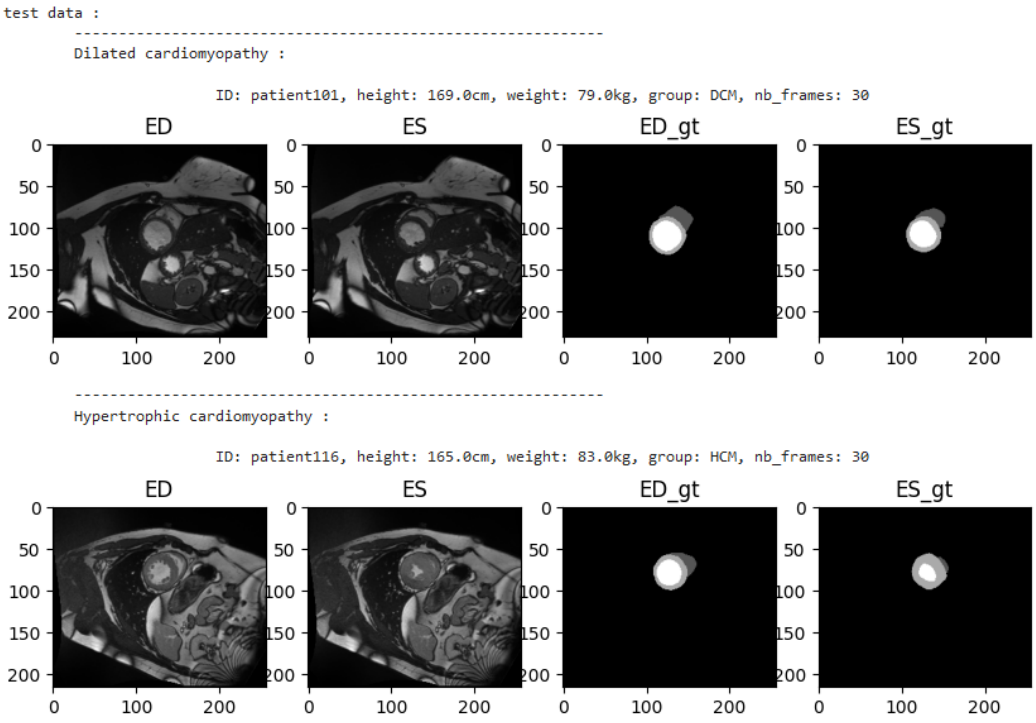

Figure 7: Example of images and metadata available for two patients in the test set

3.2 Preprocessing

The segmentation mask images are in the format height × width × depth, with integer values ranging from 0 (background) to 3, based on the segmented parts. The images do not all share the same aspect ratio or size. Moreover, the segmented region is sometimes significantly smaller than the overall image size and is not necessarily centered. This inconsistent format poses a challenge because the input data for our models must have uniform dimensions.

The first preprocessing step is to detect, for all 2D slices, the region of interest in the image and center it within a square format that is consistent across all images. To identify the region of interest in a 2D slice, we simply remove as many rows and columns as possible that contain only pixels with value 0 (the background), and then add margins to produce a square image centered on the segmentation (to avoid distorting the objects).

Next, we choose a target image size (e.g., 128×128) to which all images are resized. We use a *k-nearest-neighbor* method for resizing to ensure the pixel values remain integers in the range of 0 to 4.

As a result, our set of slices now has a standardized format. We can even visualize the segmentation masks in RGB, with each mask represented by a distinct color and black representing the background (see Figure 8).

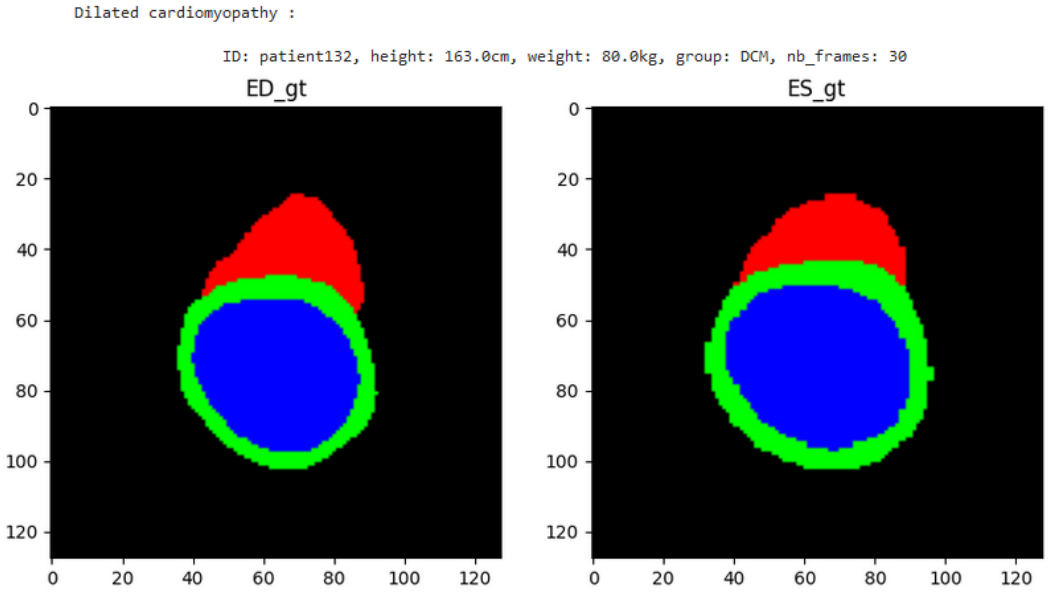

Figure 8: Example of resized images in RGB

3.3 Image Set Augmentation

After the preprocessing steps, we may question whether the dataset is sufficiently decorrelated to avoid introducing bias that could skew the training process. One potential bias could be the orientation of the images, where the right ventricle is often located at the top or the right side of the image. To mitigate this bias, we propose augmenting the dataset by generating rotated versions of the existing images, thereby increasing variability with respect to this parameter.

4 Algorithm and results

4.1 Algorithm and implementation

Our algorithm was mainly obtained from [5] and [1] .

Diffusion algorithm We didn't implement or use Conditional diffusion, nor Speed up Diffusion Models (DDIM, which may make the training faster), Classifier-Free Guidance, Blurring Diffusion, Latent diffusion, stable diffusion, etc... The implementation we did is the vanilla implementation of a DDPM, which we adapted to our data, modifying a few key parameters. We also used a specific optimized U-Net architecture which may be different from SOTA architectures for diffusion (which really depend on the size of the data and the size of the dataset), from [1].

For this part, we mainly used the classic DDPM algorithm, calculating and instantiating all the variables "By hand" without any library. We used a cosine Variance Schedule with T = 1000.

U-Net architecture U-Net architecture and parameters

- Sinusoidal position embedding to make a time vector at each condition on time (each new timestep generation)
- 2 DownSampling and Upsamping, with 3 downblocks, 3 up blocks, and 1 middleBlock
- In each block there is at least one ConvnextBlock , which is a specific Convolutional layer (cf. [3])
- 3 DownBlocks : 2 Convnext -> 1 linear attention layer (variant of self-attention but with less parameters and less power) -> 1 downsampling which is 2x2 maxpooling
- 3 UpBlocks : Input concatenated with the skip connexion-> 2 Convnext -> 1 linear attention -> 1 Up-sampling which is a 2x2 convolution
- 1 middle block : 1 convnext, 1 self-attention layer, 1 convnext
- Loss : Smooth L1 Loss
- Gradient method : Adam optimizer with learning rate = 1e-4

The notebook was made to be trained and tested on the fashionMNIST Dataset which comprised 28x28 images of clothes in black and white. We adapted the algorithm for our data by changing the model's parameters, the number of channels and the input and output size.

4.2 Results

Parameters chosen:

- Image size : 64×64 pixels
- T : 1000
- Number of epochs : 50
- Size of the batchs : 16 images
- Multiplication of features for each layer : $(1, 2, 4)$

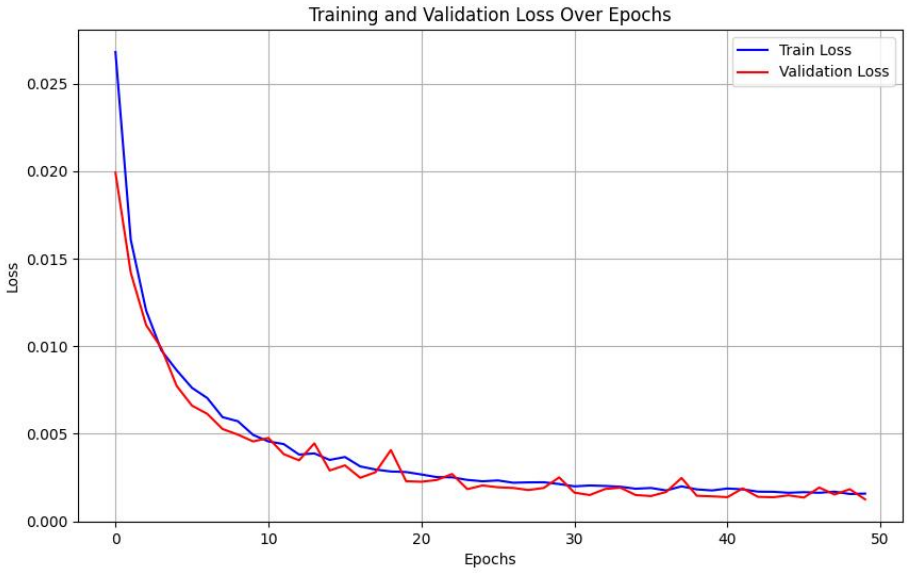

Figure 9: Evolution of training loss and validation loss over the epochs

As seen with a validation loss (computed with fixed time steps along all epochs), for now the best model after training is the last one (after 50 epochs) (as shown in Figure 9. The validation loss and training loss are similar, so we are not in the presence of over-fitting, and we could try increase the number of epochs, even if the results (as seen in Figure 10) are quite good. For better visualisation purposes, we split the images into 2 frames : one for the segmentation of the heart into 3 areas (RGB), and one for the background intensity.

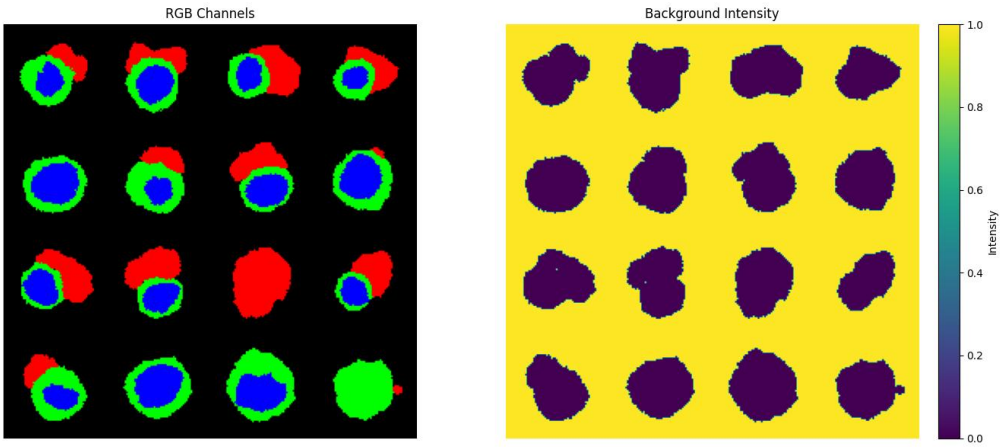

Figure 10: Sampling result from random noise

The gif format (Figure 11) shows us how the sampling process converges from random noise to the desired structures representing the heart segmentation.

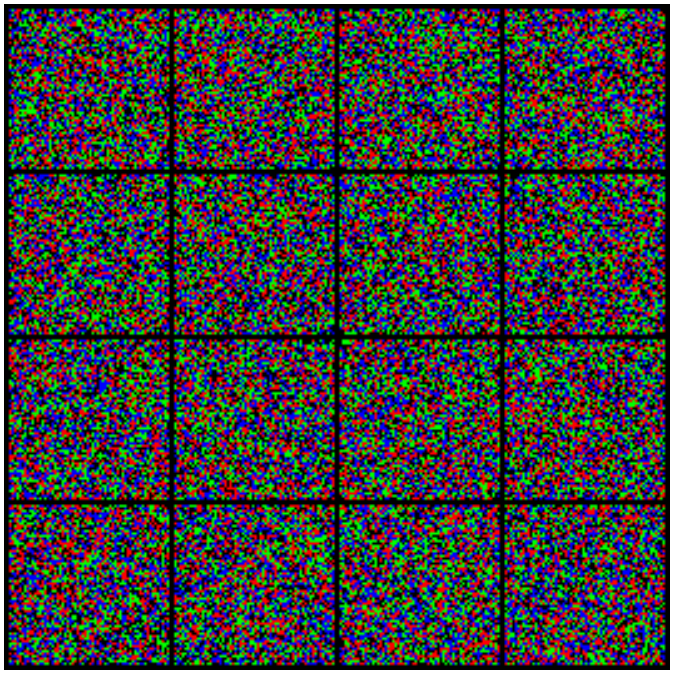

Figure 11: Sampling process - 1 frame represents a time step of 200

A next step will be to train a model with larger images, and see how the model manage to learn over more features.

5 References

References

[1] CNRS. “Diffusion”. In: *CNRS*. 2023. URL: https://www.youtube.com/watch?v=gKlo5BVpcBs.

[2] Emiel Hoogeboom and Tim Salimans. “Blurring Diffusion Models”. In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=OjDkC57x5sz.

[3] Zhuang Liu et al. *A ConvNet for the 2020s*. 2022. arXiv: 2201.03545 [cs.CV]. URL: https://arxiv.org/abs/2201.03545.

[4] Yang Song. “Generative Modeling by Estimating Gradients of the Data Distribution”. In: *yang-song.net/blog* (2021). URL: http://yang-song.net/blog/2021/score/.

[5] Lilian Weng. “What are diffusion models?” In: *lilianweng.github.io* (2021). URL: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.

[6] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. *Adding Conditional Control to Text-to-Image Diffusion Models*. 2023. arXiv: 2302.05543 [cs.CV]. URL: https://arxiv.org/abs/2302.05543.