UCLouvain

École polytechnique de Louvain

Generation and evaluation of realistic medical images with latent diffusion models for virtual clinical trials in radiation therapy

Authors: **Vincent Cammarano, Pierre Merveille**
Supervisors: **Ana Maria Barragan Montero, John A. Lee**
Readers: **Camille Draguet, Sébastien Jodogne**
Academic year 2023–2024
Master [120] in Data Science: Information technology

Abstract

In recent years, deep learning has revolutionized various fields, offering advanced tools for tasks such as generating high-quality synthetic images and performing precise image segmentation. These advancements are particularly impactful in the medical domain, where deep learning algorithms assist in identifying and segmenting tumors, diagnosing diseases,... The effectiveness of these algorithms hinges on the availability of vast amounts of high-quality training data, which is especially critical in medical imaging due to its direct influence on patient outcomes and clinical decisions.

However, generating non-synthetic data such as CT scans is resource-intensive, requiring significant financial investment and professional expertise to ensure data quality. Additionally, strict confidentiality regulations, such as the European Union's General Data Protection Regulation (GDPR), raise challenges for researchers in terms of data collection, storage, and processing. These constraints make acquiring adequate datasets for training robust deep-learning models difficult.

This thesis addresses these challenges by presenting a comprehensive workflow for generating synthetic lung CT scans with tumors. The proposed workflow integrates two key components: a variational autoencoder to derive latent representations of the CT scans, and a diffusion model utilizing a U-Net architecture trained on these latent representations. The workflow performance is assessed through both quantitative and qualitative evaluations. Furthermore, several modifications are recommended to enhance the workflow potential, aiming to provide a reliable and efficient method for generating high-quality synthetic medical imaging data.

Acknowledgements

We would like to thank our supervisors, Ana Maria BARRAGAN MONTERO and John A. LEE, for letting us work together on such an interesting subject. We are also grateful to them and Camille DRAGUET for the helpful discussions we had throughout this thesis. Thank you for your availability, supervision, and guidance, and for supporting and trusting us until the end.

We also want to thank Margerie HUET for her advice on using clusters and for telling us about the MONAI framework, which saved us a lot of time with our workflow. Thank you to Julien PIERRARD for taking the time to discuss the scans used in this thesis.

Finally, we would also like to thank our respective families and friends for their constant support during this thesis.

Contents

Chapter 1

Introduction

Over the past few years, deep learning has emerged providing more and more useful tools like generating synthetic high-quality images, segmentation tools, etc. These tools have been used in medical fields to identify and segment tumors on CT scans [20] or diagnose certain diseases such as Alzheimer's based on patient data [33].

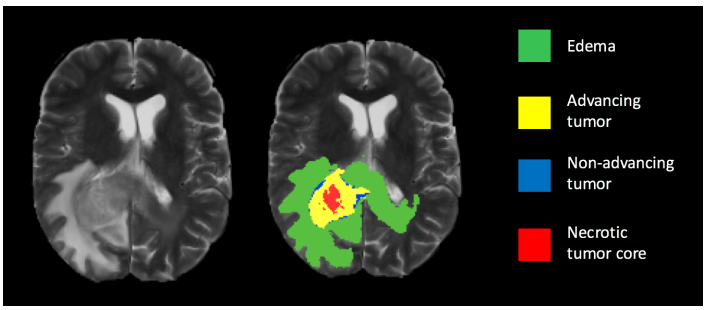

Figure 1.1: Brain Tumor Segmentation using Deep Learning. Credits: [17]

At the heart of this revolution lies the need for vast amounts of high-quality data for training robust models. Our attention is particularly focused on the medical imaging field, and the care for such data is particularly pronounced since the accuracy and confidence of deep learning algorithms directly impact patient outcomes and the clinical decisions of supervising professionals.

There are numerous imaging modalities in medical imaging such as Computed Tomography (CT), Magnetic Resonance Imaging (MRI), X-ray, Ultrasound, etc. In our case, we want to collect Computer Tomography from patients with lung tumors. CT scanning is a medical imaging technique used to obtain detailed internal body images by measuring X-ray attenuation by different tissues inside the body. Acquiring this type of modality is difficult.

Creating non-synthetic data, such as CT scans, is a task that requires a lot of resources. Indeed, the cost required to make the data itself, and the time it takes a professional to verify the quality of the sample is high. In addition to the financial and logistical challenges, researchers in the medical domain are subject to very strict confidentiality rules such as the General Data Protection Regulation (GDPR) imposed by the European Union. To respect such measures, the researchers have to take precautions when collecting, storing, and processing this sensible information. These are reasons why finding datasets of this type of modality to train deep learning models is difficult.

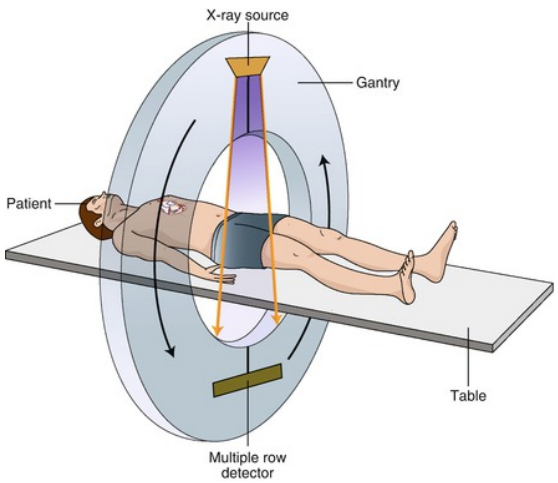

Figure 1.2: Computed Tomography (CT) principle. Credits: [18]

In order, to solve these problems companies, universities, and researchers from around the world are working on specific kinds of deep learning models called generative models. These models aim to learn the structure of images to generate images with the same characteristics. In the literature, certain types of generative models stand out from the others: Generative Adversarial Networks (GAN), Transformers, Autoencoders, or Diffusion models. In our specific case, the diffusion models were proven to be the best for the generation of high-quality samples [10, 22]. The other types of models have major drawbacks, such as generating images with little variation between them (mode collapse), or being resource-hungry and requiring a lot of training data.

This thesis presents the development of a comprehensive workflow for the generation of lung CT scans with tumors. Advanced deep learning tools were leveraged to design and optimize each step of the process, from model selection and training to rigorous performance evaluation and inference of results. The thesis includes an in-depth analysis of the deep learning techniques employed, along with methodologies to enhance the accuracy and efficiency of generations. The results obtained demonstrate an advancement in the automated generation of lung CT scans with tumors, with substantial potential applications in the medical field.

This thesis is divided into 7 chapters after the introduction. Chapter 2 covers the theoretical aspects of the various deep learning models/tools used. The third chapter explains the principle of the diffusion model which is at the heart of the project. The fourth chapter consists of a review of the literature on the subject. The fifth chapter covers the methodology used, taking up from start to finish what we have done to obtain our generative model evaluated in Chapter 6. Chapter 7 contains a list of ideas for improvements and applications. Finally, Chapter 8 includes the conclusion of the thesis.

Chapter 2

Deep Learning fundamentals

In recent years, the field of Deep Learning has seen major developments in many artificial intelligence tasks. We have seen advances in natural language processing with the arrival of LLMs such as GPT-4 [38]. Computer vision has also seen advances with image generation models such as Dall-E [41] and, more recently, video generation models such as Sora [5]. These improvements are due to several factors, including free access to a wide range of data sources, the development of new, more complex architectures, and increased computing power.

In this section, we will introduce some fundamental Deep Learning concepts that we need to develop an image generation model using the stable diffusion principle.

2.1 Artificial Neural Networks

Before talking about artificial neural networks, it is important to define the concept of artificial neurons. The artificial neuron is inspired by a simplification of neurons in the brain.

Artificial neurons comprise several inputs representing the signals that neurons receive, a processing unit, and a single output. Each of the inputs is associated with a weight that represents the level of importance that the input has. The processing unit performs a weighted sum of the inputs multiplied by their corresponding weights. After calculating the weighted sum, the neuron applies an activation function. The activation function introduces non-linearity into the neuron's output, enabling neural networks to learn complex models. The final output is the result of this activation function. The objective of these artificial neurons is to modify their weights via a learning algorithm so that the resulting value is the one expected based on the given input.

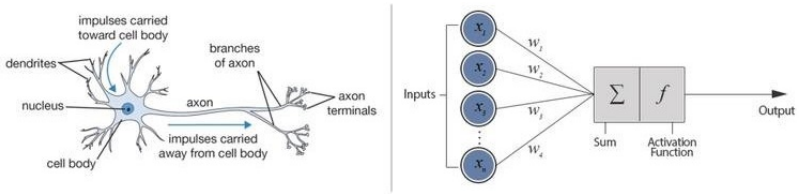

Figure 2.1: biological neuron vs artificial neuron. Credits: [34]

These artificial neurons can be combined to create an Artificial Neural Network also called feedforward neural networks. These combinations of neurons form layers. Like artificial neurons, an Artificial Neural Network is made up of an input and output layer. However, to this, we add hidden layers. The hidden layers are made up of a set of artificial neurons. The artificial neurons of the first layer take the initial input and the output of the neurons of this layer becomes the input of the next layer up to the output layer. It is important to note that in this model type, each artificial neuron is connected to all the inputs and the neurons in the next layer. An ANN with at least 2 hidden layers is called a Deep Neural Network.

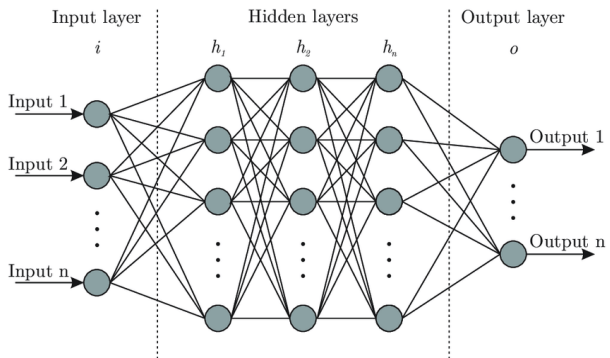

Figure 2.2: Illustration of an Artificial Neural Network. Credits: [4]

The algorithm for learning the weights of the various artificial neurons follows the concept of function minimization in mathematical analysis. The objective is to minimize a cost function calculated as a function of the values predicted by the model and the original values. For example, a cost function often used for regression tasks is Mean Square Error (MSE).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.1}$$

Learning the weights then minimizes the error between the values predicted by the model and the original values. This process is called back-propagation. Weights are then updated by calculating the gradients of the loss function with respect to the weights. These gradients, multiplied by a coefficient (learning rate), are back-propagated, updating the weights. This process is then iterated until the cost function converges.

Once the function has converged, the model is then evaluated based on a set of data not used during the weights training. The original data is separated into a training set and a testing set. This evaluation aims to measure the generalization of the model on new data. If the results of this evaluation are not good, this means that the model overfits the training data and can therefore only be used on this data. In this case, several solutions can be envisaged, such as modifying the model's meta parameters (number of layers, etc.), changing the optimizer (Subgradient Descent, etc.), or using other cost functions.

2.2 Convolutional Neural Networks

Although Deep Neural Networks can be used to model complex patterns in data, they do have their limitations. This is because DNNs have limitations with multi-dimensional and spatial data such as images. A DNN using an image as input would need to have the values of each pixel, which would greatly increase the number of weights present in the model. Furthermore, as each artificial neuron in a layer receives all the inputs from the previous layer, the model cannot identify local patterns specific to an image.

To overcome these limitations, researchers have developed a new architecture. This new architecture was inspired by the animals' visual system, which scans the environment to create an image. This architecture uses convolutions between an image and a small-size filter. The filter achieves the same local operations uniformly across the entire image, a property known as shift-invariance.

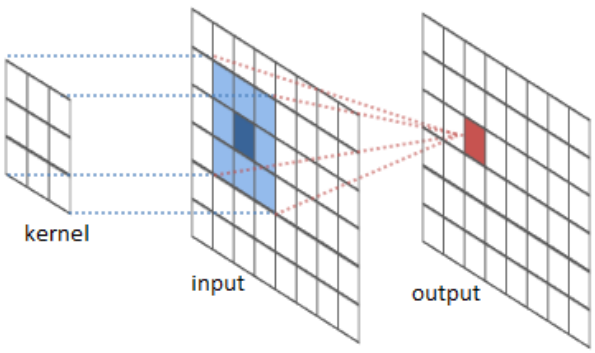

Figure 2.3: Illustration of convolution. Credits: [26]

The output grid in Figure 2.3 is then called a feature map because convolution aims to extract features from the input. A neural network model using convolutional layers is called Convolutional Neural Network (CNN).

The term convolution first appeared in 1989 when LeCun et al [31] built a convolutional neural network for handwritten zip code recognition, which is the original version of LeNet.

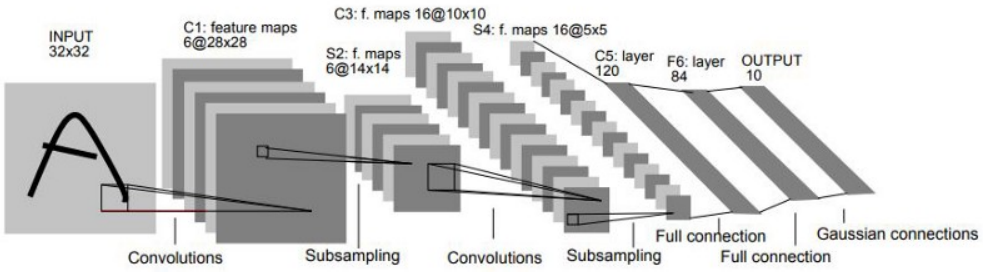

Figure 2.4: Architecture of the CNN called LeNet-5. Credits: [31]

The CNN architecture shown in Figure 2.4 consists of a first part with combinations of convolutional layers and subsampling layers and then a fully connected part (DNN).

As with DNNs, convolutional layers work with a weighted sum and an activation function. However, the weights are different. In a convolutional layer, the filter or kernel is made up of a matrix. The elements of this matrix are the weights. This weighted sum is illustrated in Figure 2.5. The neural network units are no longer connected to all the inputs but only to certain regions, resulting in fewer connections and therefore fewer weights.

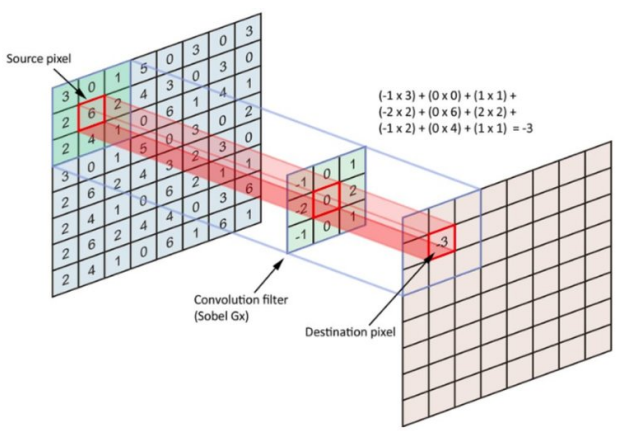

Figure 2.5: Illustration of convolutional layers with a 3x3 2D-kernel and 0 padding. Credits: [35]

The size of the feature map depends on several parameters:

- The filter size which is the dimension of the matrix.
- The stride is the jump made when shifting the filter.
- The padding, is the process of adding extra pixels or values around the border of an input image or feature map before applying a convolution operation.

The subsampling layer or pooling layer is a type of layer used to reduce the spatial dimensions (width and height) of feature maps while retaining important information. This dimension reduction reduces the complexity of the subsequent layers, making the model more efficient. This reduction is achieved via a pooling window which shifts, depending on the stride defined, in the same way as for the convolutional layer. The most commonly used pooling techniques are Average Pooling and Max Pooling [55]. The Average Pooling takes the average value of all features in the pooling window, which has the effect of smoothing the values. The Max Pooling takes the maximum value of all features in the pooling window.

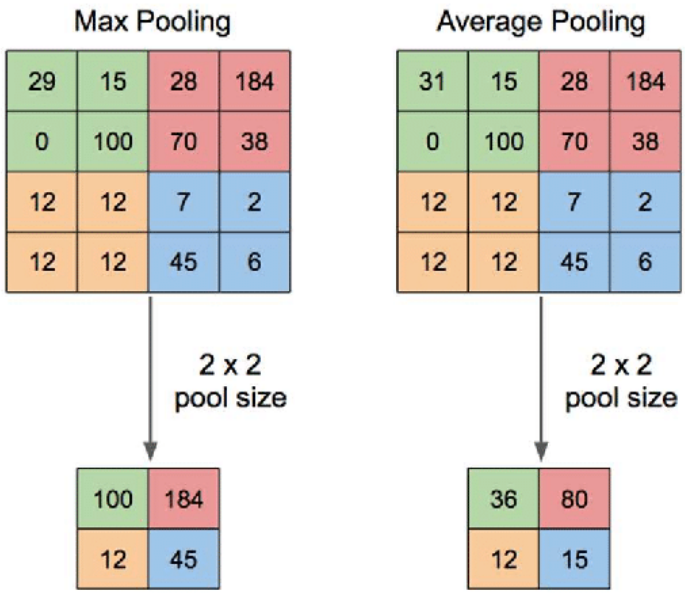

Figure 2.6: Illustration of Max Pooling and Average Pooling. Credits: [32]

2.3 ResBlock

In deep learning, it is often said that the deeper a network is, the better it will perform. However, the complexity of a neural network gives rise to other problems, such as gradient vanishing and gradient explosion. Gradient vanishing occurs when, in order to update the weights, the magnitude of the gradients decreases sharply until the weights no longer change. Gradient explosion is the opposite process.

In 2015, Kaiming He et al [21] developed a new architecture for Image Recognition called ResNet. This new type of architecture is based on new type of block called ResBlock to avoid the vanishing gradient. This new type of block uses residual/skip connections to learn residual functions that refer to the input layer.

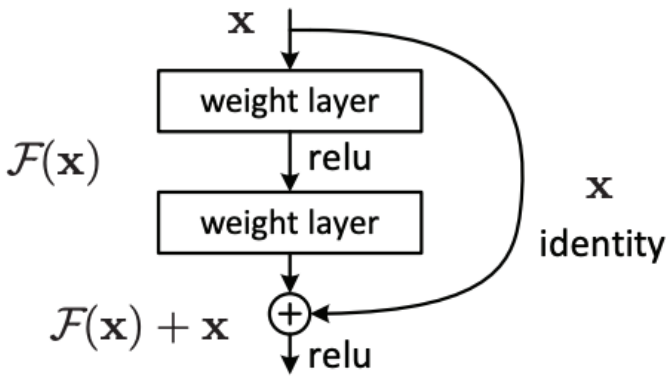

Figure 2.7: Concept of ResBlock. Credits: [11]

The output of the block shown in Figure 2.7 is no longer a set of hidden layers representing the $\mathcal{F}(\mathbf{x})$ mapping. Instead, the block mapping is transformed into $\mathcal{H}(\mathbf{x}) := \mathcal{F}(\mathbf{x}) + \mathbf{x}$. The objective is no longer to learn the complete output but the following difference: $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$. The mapping $\mathcal{F}(\mathbf{x})$ is called the residual function.

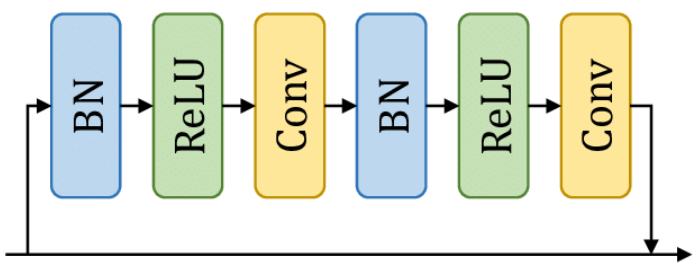

Figure 2.8: ResBlock for images. Credits: [59]

Figure 2.8 shows the architecture of a ResBlock using convolutions. There are two combinations of three layers. Firstly, a layer called Batch Normalisation. Next, an activation layer with the ReLU function. And finally, a convolutional layer. Batch normalization applies a transformation that keeps the mean output close to 0 and the standard deviation of the output close to 1.

2.4 U-net

In 2015, Olaf Ronnenberger et al [45] developed a new CNN architecture called U-Net. This architecture was originally developed for image segmentation tasks in the biomedical field. This new approach has had a bombshell effect on computer vision research.

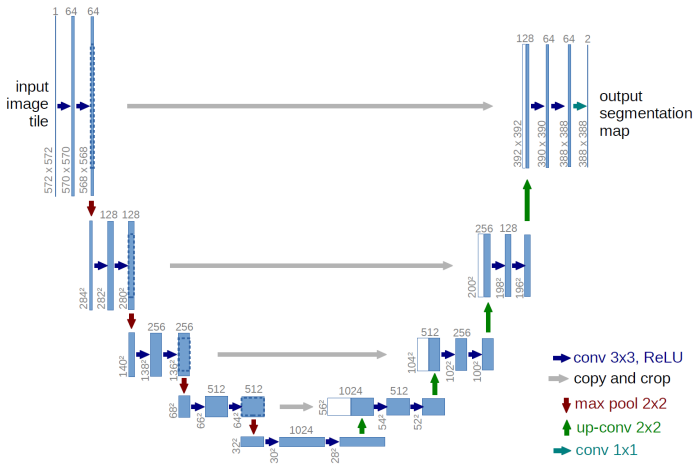

Figure 2.9: U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. Credits: [45]

The name U-Net comes from the shape of the U-shaped architecture. The architecture is made up of two parts:

- The first part is made up of convolutional layers and Max Pooling layers to extract features from the image and extract its context. This section reduces the resolution of features down to the bottleneck of the model. This part is called the contracting path.
- The second part, the expansion path, performs upsamplings and convolutions symmetrically to the first part. This section also incorporates skip connections, allowing it to combine the matching cropped feature map from the contracting side of the network. These skip connections enable precise localization of the different elements in the image by regaining the spatial information that was lost during the downsampling process.

This architecture has been adapted for 3-dimensional images by Özgün Çiçek et al [60].

2.5 Autoencoder

In the early 1990s, following a series of research papers [30] [37] [9], a new neural network architecture was developed. This architecture is called the Autoencoder (AE). This architecture aims at learning a compressed representation of the input data and then reconstructing the original image based on this representation without supervision. This architecture has found wide applications in areas like data compression, anomaly detection, denoising, and more.

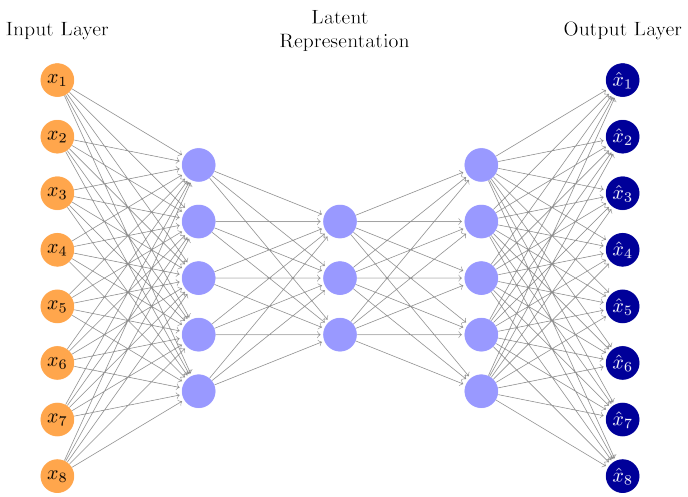

Figure 2.10: Illustration of Autoencoder. Credits: [43]

Figure 2.10 shows an example of Auto-Encoder architecture. This symmetrical architecture comprises two distinct components (DNN): Encoder and Decoder.

- Encoder: This first component is used to compress the data present in the input layer into a compressed representation, also known as a latent space representation. This compression is carried out using a series of layers containing fewer and fewer artificial neurons until the desired size is reached. The output layer of this encoder is also called the AE bottleneck.
- Decoder: This second component does the opposite. It tries to reconstruct the original input data using the latent representation. To do this, the decoder comprises a series of layers with more and more artificial neurons to obtain the size of the original data space.

This model is trained compactly, i.e. the Encoder and Decoder form a single model. Autoencoders are trained using a reconstruction loss function, such as mean squared error (MSE). The goal of training is to minimize the difference between the original input and the reconstructed output. During training, the network adjusts the weights in the encoder and decoder to achieve better reconstruction accuracy. This architecture has been adapted for use with multi-dimensional data such as images. To do this, the Encoder and Decoder are replaced by CNNs.

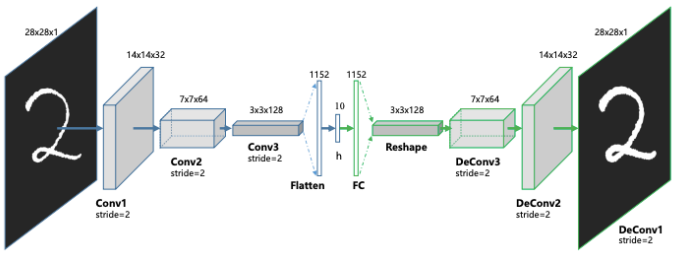

Figure 2.11: Illustration of Autoencoder with CNNs. Credits: [8]

2.6 Variational Autoencoders

Traditional autoencoders have their limitations. This type of model is deterministic, which means that it produces a fixed output based on a given input. Using this model with a small variation in the given input does not give a result related to the previous output. Another limitation is the weak generalizations that this model has on new data even if this varies little compared to the training data.

In 2013, Kingma and Welling [29] developed the Variational Autoencoder (VAE) offering a new approach to autoencoders by incorporating probabilistic elements into the latent space. This approach addresses some of the limitations of traditional autoencoders and allows VAEs to be used for a wider range of applications, particularly in generative modeling.

A major difference with conventional AE is the encoder output. In VAE, the encoder output represents the mean and standard deviation of a Gaussian distribution. After recovering these two elements, the model creates the latent representation based on a sampling of this probabilistic distribution.

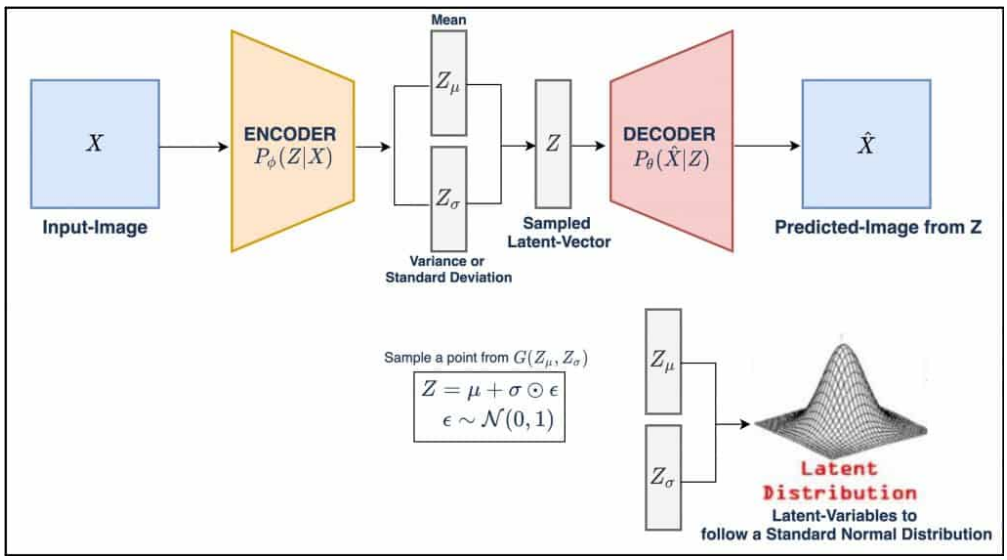

Figure 2.12: Illustration of Variational Autoencoder. Credits [46]

In addition, VAEs introduce an additional loss component based on the Kullback-Leibler (KL) divergence. This divergence measures the difference between two probability distributions. The aim is to make the latent space as smooth and structured as possible. To achieve this, VAEs add to the loss reconstruction, the KL divergence between the learned distribution and a standard Gaussian distribution.

The KL divergence between two Gaussians can be obtained using the following equation:

$$\begin{aligned}\mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}\left(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0\right), \mathcal{N}\left(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\right)\right) = \frac{1}{2}\Bigg(& \operatorname{Tr}\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0\right) \\ & - d + \left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0\right)^T \boldsymbol{\Sigma}_1^{-1}\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0\right) \\ & + \log \frac{\det \boldsymbol{\Sigma}_1}{\det \boldsymbol{\Sigma}_0}\Bigg).\end{aligned} \tag{2.2}$$

2.7 Self-Attention Block

In Natural Language Processing, Vaswani et al [51] published a paper in 2017 about Self-attention-based architectures for translation tasks. Self-Attention mechanism allows a model to focus on different parts of the input data by weighing their relevance.

Self-attention works by creating a set of weights that represent the relevance or importance of each part of the input to other parts. Mechanism can be divided as follows:

- Input data is transformed into three different matrices, often using linear projections. These three transformations are called Query, Key, and Value. The Query represents the current position in the sequence and asks what other positions have relevant information. The Key represents all positions in the sequence and contains the information that queries can match against to find relevant data. The Value contains the actual information from the sequence that will be used for the attention calculation and output.
- The attention score is then calculated by multiplying the points between the Queries and the Keys. This dot product is then scaled and softmax normalized to ensure that the sum of the scores equals one. The normalization step allows the model to focus on the most relevant parts of the input.
- The final output is created by summing the Values weighted by these attention scores. This process allows each point in the output to account for every other point in the input, helping the model understand the broader context.

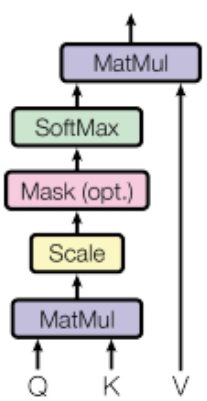

Figure 2.13: Self-Attention mechanism. MatMul represents the dot product. The filer layer is optional. Credits: [51]

Based on this architecture, several studies have tried to incorporate this mechanism with CNNs to manage images [53]. Usually, this mechanism is used in conjunction with a CNN or to replace part of a CNN that retains its structure. In 2021, Alexey Dosovitskiy et al [14] managed to incorporate this mechanism into computer vision tasks without the need for CNN.

In computer vision, the self-attention mechanism is applied to the feature maps of a CNN. This allows the network to focus on important image regions while suppressing irrelevant information. The Self-Attention mechanism is used for generative models by helping the generator to focus on relevant image regions.

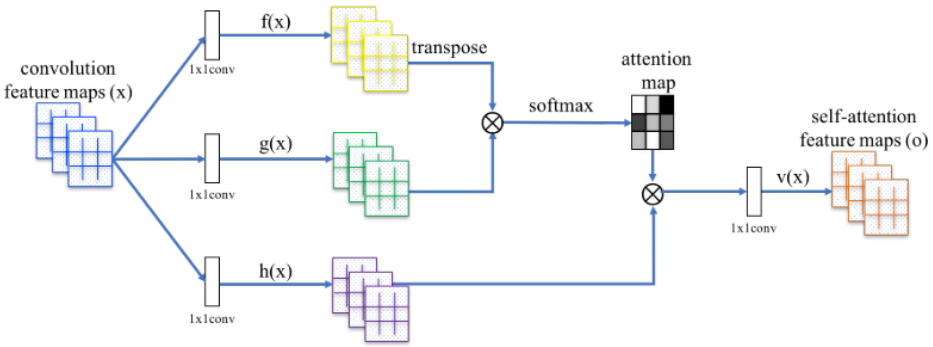

Figure 2.14: Self Attention in Convolutional Neural Networks. Credits: [2]

Chapter 3

Diffusion model

This chapter introduces the model that has restarted the race of generating images. The original diffusion probabilistic model (DDPM) was introduced in 2015 by [47], and its essential idea came from Thermodynamics. Five years later, a research team at the University of Berkley presented the paper [22], where they applied this principle to images.

The core idea behind diffusion models is to simulate the operation of adding noise to data and then learn to reverse this process to reconstruct the original data. This method is divided into two parts: the forward process and the reverse process.

3.1 Forward Process

In the forward process, Gaussian noise is progressively added to the data in a series of steps. The image is then noisier and noisier until a random distribution is obtained.

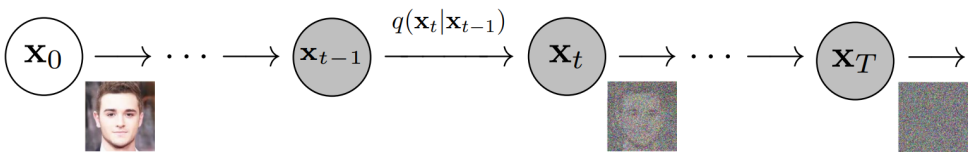

Figure 3.1: Forward Process in Diffusion Model. Credits: [22]

This process is modeled as a Markov chain. Markov models are a class of probabilistic models. This type of model is often represented as a set of states linked by transitions. These transitions represent the probability of passing from one state to another. A Markov chain has an important assumption. The probability of being in a state at a time t depends only on the state at a time $t-1$.

$$P\left(x_t \mid x_1, \ldots, x_{t-1}\right) \approx P\left(x_t =\mid x_{t-1}\right) \tag{3.1}$$

The Figure 3.1 shows a conditional distribution $q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)$. This distribution gives the probability of having X_T by having X_{t-1}. Mathematically, the forward process is defined as follows:

$$q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right) := \prod_{t=1}^{T} q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right) \tag{3.2}$$

$$q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right) := \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right). \tag{3.3}$$

Equation 3.2 represents the probability of having any state with the initial state as the prior knowledge. It is obtained by multiplying the conditional probabilities of the previous states. For example, The probability of obtaining the state $\mathbf{x}_2$ is equal to the probability of $\mathbf{x}_1$ if we have $\mathbf{x}_0$ times the probability of $\mathbf{x}_2$ if we have $\mathbf{x}_1$.

Equation 3.3 means that the probability distribution of $\mathbf{x}_t$, given $\mathbf{x}_{t-1}$ is a Gaussian (Normal) distribution. The parameters for this distribution are:

- The mean is equal to $\sqrt{1-\beta_t}\mathbf{x}_{t-1}$. This suggests that the expected value of $\mathbf{x}_t$ is a scaled version of $\mathbf{x}_{t-1}$.
- The covariance matrix is $\beta_t \mathbf{I}$. It is the Identity matrix scaled by the factor β_t.

The parameter β_t controls the amount of noise or randomness introduced at each step. A higher β_t leads to more noise and greater uncertainty and a lower β_t means the transition is more deterministic, with less deviation from the expected trajectory. This forward process is defined by $\beta_1, \ldots, \beta_T$ called the noise scheduler.

A notable property of the forward process is that it admits sampling $\mathbf{x}_t$ at an arbitrary timestep t based on $\mathbf{x}_0$. Using the notation $\alpha_t := 1-\beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$,

$$q\left(\mathbf{x}_t \mid \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, \left(1-\bar{\alpha}_t\right)\mathbf{I}\right)$$

3.2 Reverse Process

As its name suggests, the reverse process involves training a neural network to reverse the forward diffusion process, ultimately recovering the original data from noisy input.

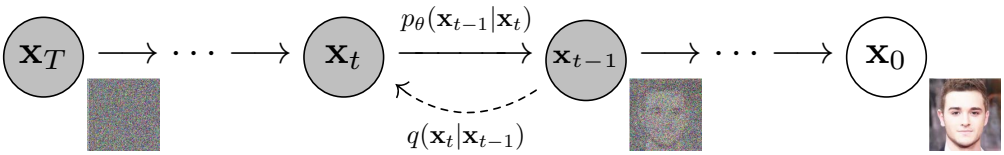

Figure 3.2: Reverse Process in Diffusion Model. Credits: [22]

A Markov Chain also represents this process. This Markov chain starts with $p(x_T) = \mathcal{N}(\mathbf{x}_t; 0, \mathbf{I})$. The above equality means that the Markov chain starts with standard Gaussian noise. Mathematically the reverse process is written like this:

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \tag{3.4}$$

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{3.5}$$

The equations below depend on a different probability distribution $p_\theta(.)$. The θ symbol means that the distribution depends on certain learnable parameters. As a result, the mean and covariance matrix of the Gaussian distribution are also influenced by this parameter. By learning this parameter, it is possible to identify the noise that has been added between two states during the forward process.

3.3 Training process

Training a diffusion model differs from the classical processes seen in deep learning. It consists of minimizing the difference between the expected noise added a timestep t and the true noise added. A training step is done for a randomly generated timestep t. The training algorithm is shown below:

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: 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

Figure 3.3: Training Process for Diffusion Model. Credits: [22]

Initially, the algorithm generates a timestep t from a Uniform distribution and noise ϵ from a standard Gaussian distribution. Based on these elements and the forward process property, x_t can be generated based on x_0. Consequently, the model is trained by minimizing the error between the true noise ϵ and the noise predicted by the model ϵ_θ at time x_t.

Chapter 4

State of the art

This chapter delves into a comprehensive review of various papers that have developed and presented complete frameworks centered around the diffusion principle. While there has been substantial investigation into different diffusion frameworks, many of these efforts have not focused on the medical domain, particularly in the context of CT or MRI scans.

Following the paper on the denoising diffusion principle applied for image generation, numerous research teams began to explore this specific process. The first notable paper in this context comes from an OpenAI team, published a few months later. Their results provide a concrete assessment of potential improvements to the architecture of diffusion models, demonstrating superior image generation performance compared to GANs networks [10].

Rombach et al. [44] then published new architectures using the diffusion process called the Latent Diffusion Model (LDM). This work provides a concise view of leveraging latent space for image synthesis. This paper focuses on generating high-quality images from popular datasets such as CelebAHQ and LSUN-Beds. Pinaya et al. [40] explored latent diffusion models for generating brain CT images, which is directly related to this thesis.

Finally, the most recent and comprehensive framework for image generation presented is GenerateCT [19]. This framework generates 3D chest CT volumes by combining three deep-learning models, each assigned specific tasks, thereby representing the best and most complete current approach in the field.

4.1 Diffusion Models

The published diffusion models, presented in Chapter 3, could not produce high-quality samples. However, by improving the architecture of the state of the arts of diffusion models, the team from OpenAI, that published [10], has shown that they can perform image inference at 512x512 resolution.

The first improvement made is a discovery from [36] where it is proved that a fixed variance for a small diffusion step is not optimal, by making it a learning parameter during the backward process reduces the number of diffusion steps needed to denoise an image. Many experiments have been carried out to optimize existing or new layers. These experiments have focused on exploring the number of attention heads and their size, re-scaling the residuals connection, and adding residual block upsampling/downsampling operation on the activation function.

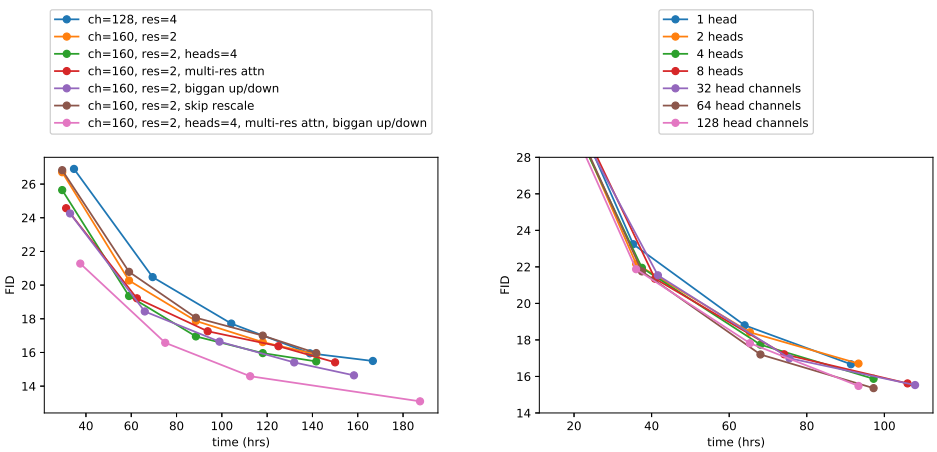

Figure 4.1: Experimentation of various architecture changes, showing FID as a function of wall-clock time. FID evaluated over 10k samples instead of 50k for efficiency. Credits: [10]

It is possible to observe, that the evaluation metrics (FID) are improved with time when augmenting the number of channels and/or the number of heads in the attention mechanism.

Another improvement from them was already in place for the above experiments [36], it consists of a new layer called adaptive group normalization (AdaGN). This layer performs a linear operation on the group normalization operation [56] of the activation function of different hidden layers.

4.2 Latent Diffusion Models

Latent Diffusion Models is a new kind of combined architecture explored by Rombach et al [44] in 2022. At that time, the diffusion of high-resolution pixels was still far from perfect due to the high dimension of the images in the initial space. The aim is to apply the principle of diffusion in a latent space and then transform the latent representation into an image.

The first block of the architecture is an autoencoder model, whose role is to reduce the dimensionality of the input images in a latent space, where it gains a representation of the images that is equivalent to the originals. The second block reproduces a U-net model used to learn the diffusion process on the latent space of the autoencoder.

This approach offers multiple advantages: Firstly, training the diffusion model is significantly more cost-effective. Secondly, since the autoencoder captures the features of the original image inside the latent features, it alleviates the work of the U-net architecture for the diffusion process. Thirdly, once the autoencoder is trained, the latent space provided by the encoder block can be used to train multiple models with different purposes.

The losses used during the training of the autoencoder are a combination of training loss. A perceptual loss [57], patch-based [24] and adversarial objective [15]. The purpose of this combination is to guarantee the quality of the reconstruction images, it bypasses the blurriness that can happen when training this kind of model by only relying on mean squared error or mean absolute error. In addition to this combination, they added a Kl-divergence regularisation term, because it helps to regulate the variance of the latent space.

The result of what has been stated before is that there is no need for an aggressive reduction of the original space to the latent space to achieve good reconstruction. While previous work was reducing the dimensionality of the image to 1D [42], the outcome of their experiences is that reducing the dimensionality by a factor of 4 is sufficient. The main advantage of this is that the latent representation learned contains much more information about the original data space. Figure 4.2 shows a diagram of this architecture.

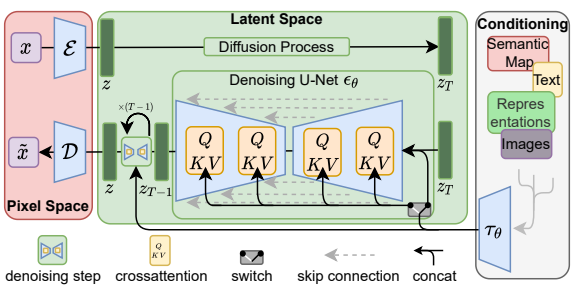

Figure 4.2: Representation of the LDM architecture. Credits: [44]

In addition to this framework, they added the conditioning mechanism to the diffusion model. This is done by modifying the structure of the U-net by adding layers of classical self-attention as defined in [51].

Following this paper, Pinaya et al. [40] investigated this concept of latent diffusion models on the UK Biobank dataset which contains 31 740 MRI images of the brain. Their goal is the same as in this thesis, namely creating and making available to the scientific community large sample of synthetic data. In their study, the compression factor is 8 which make their original input from 160x224x160 to 20x28x20 resolution in the latent space. This is the only difference from the previously presented papers, the rest of the architecture and losses are the same. The trained diffusion model is conditioned on different attributes that were available in the data, such as age, gender, ventricular volume, and brain volume. The conditioning operation is done as defined by the previous paper.

What is interesting about the result of this research, is the comparison with other generative models in the evaluation procedure. They compared their LDM model with the original images and two other baselines, a VAE-GAN, and a LSGAN, which gives a concrete impression about how well the model performs. The authors also noticed that LDMs are easier to train and capture efficiently the details of the images compared to the GANs architecture well known to have an unstable training and mode collapse of the generated latent space. In order to compare these models, multiple metrics have been used: the Fréchet Inception Distance (FID), MultiScale Structural Similarity Metric (MS-SSIM), and 4-G-R-SSIM. These metrics are computed as the average of 1000 samples.

	FID ↓	**MS-SSIM ↓**	**4-G-R-SSIM ↓**
LSGAN	0.0231	0.9997	0.9969
VAE-GAN	0.1576	0.9671	0.8719
LDM	**0.0076**	**0.6555**	**0.3883**
LDM + DDIM	0.0080	0.6704	0.3957
Real images	0.0005	0.6536	0.3909

Table 4.1: Quantitative evaluation of the synthetic images on the UK Biobank. Credits: [40]

Table 4.1 shows that LDMs outperform the baseline model in both performance and image quality, making them a strong alternative to GANs. Additionally, the LDM + DDIM model improves upon DDPM by reducing diffusion steps by 500 without sacrificing accuracy [48]. Due to its high performance, over 100,000 synthetic data samples have been made publicly available for the scientific community.

4.3 GenerateCT

GenerateCT published in late 2023 by Hamamci et al [19] is the most elaborate and up-to-date generative framework in the medical domain. The dataset used for this research is composed of 49 138 CT volumes of the chest region, with a resolution of 512x512 and slices going from 100 to 600. The metadata of these CTs are composed of different patient information such as age, sex, and something they called impression which is information about radiological reports that is used for the conditioning. The Hounsfield units of each voxel are clipped in the range of -1000,1000.

As mentioned in the introduction of this chapter, GenerateCT is a framework of three components. The first one called CT-ViT is a vision transformer that encodes 3D CT into tokens so that they can train models in high resolution. The second one is also a transformer whose role is to align the embedded 3D CT tokens with the conditioning information from the radiological report. These two models are used for the generative part conditioned by text. The third piece of this framework is a cascaded diffusion model [23] to upsample the images generated by the previous layer from 128x128xN (N is an arbitrary number of slices) to 512x512xN. Here is an example of the framework:

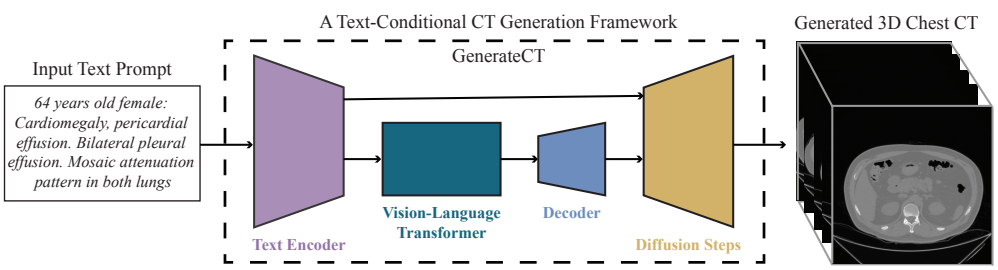

Figure 4.3: GenerateCT is a cascaded framework that generates high-resolution and high-fidelity 3D chest CT volumes based on medical language text prompts. Credits: [19]

During the training stage, each component of the GenerateCT framework was trained independently with 3D CT volumes of size 512x512x201. The training of all models is done on 8 Nvidia A100 GPUs. The CT-ViT model was trained for 100 000 epochs, the second model for 500 000 epochs and the upscaling cascading diffusion model for 275 000 epochs. The training for each of the models took a week to complete. A quantitative evaluation and an assessment by two experts in the medical domain have been realized. Were the experts able to classify synthetic versus real data? The result of that experiment is that the radiologist noticed high reliability between the conditioning and the synthetic CT. Following these results, more than 20,000 synthetic data were generated for the scientific community.

4.4 Other papers

This section includes other interesting research papers on image generation using diffusion principle in the medical field:

- **GH-DDM by B. et al. [58]**: This paper introduces a hybrid denoising diffusion model that combines U-Net architectures with transformers. It separates the diffusion process into textual and structural stages and incorporates class-conditional generation, specifically for lung CTs.
- **Khader et al. [27]**: This study presents a 3D latent diffusion model using a VQ-GAN as the autoencoder and the U-Net architecture for diffusion, similar to the approach in this thesis.
- **CoLa-Diff [25]**: This framework conditions the latent space and diffusion network with anatomical structure masks for multi-modal MRI synthesis.
- **Peng et al. [39]**: This paper focuses on generating realistic 3D brain MRIs by combining 2D generated slices, with each slice conditioned on the previously generated ones.

Chapter 5

Methods

This chapter defines the different methods used and developed throughout the whole project. The first section defines the data used and the processing carried out to train the different models. The second section describes the framework used to develop the generative model workflow. The fourth and fifth sections cover the architectures making up the generation process, with the configurations evaluated as being the best. The final section defines the inference process for a newly generated image.

5.1 Data Preprocessing

The data used for the project comes from a public repository [1]. This dataset contains 422 3D volumes of CT scans from patients, each of them having non-small cell lung cancer (NSCLC). For each patient, a 3D scan of the patient's chest, an RTSTRUCT file containing the contours of different parts of the anatomy, and a file containing the segmentation of the organs and the principal tumor of the patient, are available. This principal tumor volume is called GTV-1 (Gross Tumor Volume).

Every file is stored inside Dicom files, which is a common format used in the medical domain to transmit information. This format can store information such as the patient's age, the patient's sex, the scanner configuration, tumor stage... While Dicom is great for visualizing or transmitting the data, it is not the best format to feed deep learning models. This is why the dicoms have been converted into Nifti file format which is also a common extension used by medical professionals. In addition to the conversion to Nifti, the contours in the RTSTRUCT file have been converted to full mask with fill. Moreover, the names of the various contours (lungs, GTV-1, GTV-2, etc.) have been standardized because several names existed in the dataset for the same element. This initial pre-processing was carried out using the MIRO laboratory implementation.

To ensure the integrity of future experiences, great care was taken in the data cleaning process. Because there is a limited amount of data, it is necessary to check its quality to ensure that poor samples do not interfere with the training process. Lacking expertise in medical terminology, a visual review of each CT scan was not conducted. Instead, it was assumed that the scans were of good quality since they were sourced from a professional research institute. However, it was verified that each patient had the three required components files (CT, STRUCT, SEG). The patient LUNG-128 was removed because the segmentation file was missing. Upon further investigation in the public repository, it was discovered that this patient had undergone surgery to remove the tumor, leading to the decision to exclude him from the dataset.

The second step in data pre-processing is to import the data into the Python framework. The scans are then loaded into a Pytorch customer dataset from the MONAI framework[1]. During loading, the scans pass through transformation layers available from the MONAI framework[2]. Here is the list of transformations used:

- **LoadImaged**: This transformation loads the CT scans and contours based on the paths given.
- **MergeSegmentationd**: This is a custom transformation class that is made to retrieve each of the masks into individual matrices. The masks of secondary tumors are then merged. The output of this transform is a dictionary composed of the CT, the GTV-1, and the concatenated masks. This transformation will be useful for a future project to condition generation based on tumor masks.
- **EnsureChannelFirstd**: This transformation adds one channel to the tensors so that it has the right dimension to train the models.
- **ScaleIntensityRanged**: This transformation scales the data intensity in the range of -1000 to 1000 Hounsfield units. The selected range is inspired from [19].
- **Resized**: This transformation resizes the original dimension of the CT scans by nearest neighbor interpolation into 128x128x128 resolution.

The transformations applied to the scans are carried out each time they are selected in the DataSet, which takes time. In order not to waste too much time, these transformations were applied once and the result was saved in new Nifti files. The DataSet class retrieves images one by one. However, deep learning models are trained using minibatch data. In addition, the data needs to be reshuffled at each epoch to reduce model overfitting. This

[1] `https://docs.monai.io/en/stable/data.html`

[2] `https://docs.monai.io/en/stable/transforms.html`

is why training is carried out using a DataLoader, which also speeds up data recovery through multi-processing.

As explained in Section 2.1, the data was separated into a training set and a testing set. The training set contains 95% of the images in the dataset, corresponding to 400 CT scans. This ratio has been chosen to have enough data for the training stage.

5.2 Latent Diffusion Model

The technical implementation of the latent diffusion model relies on the MONAI framework [16]. This framework is a Python open-source project dedicated to the medical domain, aimed at accelerating the research, development, and implementation of deep learning applications. For students, such an environment provides extensive resources and documentation to learn and develop knowledge in depth. Additionally, it allows for a focus on the core problem rather than spending weeks coding well-known architectures. Another advantage is that using a higher-level resource makes the technical implementation more readable and reusable. The two architectures being used (VAE and U-net), implemented by MONAI, are based on the papers [44] and [40], which are presented in Section 4.2.

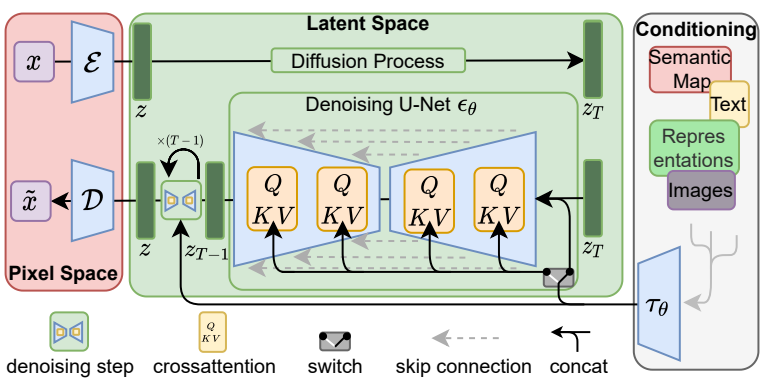

Figure 5.1: Representation of our LDM architecture. Credits: [44]

The above scheme is an abstract representation of the whole architecture. Where x is the input CT scan, the encoder $\mathcal{E}$ of the VAE downscales the input to the latent space z. The diffusion process explained in Section 3 is then applied to the scan in latent space. It is then upscaled by the decoder $\mathcal{D}$ of the VAE to output a CT scan in the same dimension as the original one.

5.3 VAE

This section describes the first model of the pipeline, as well as its architecture and training process. This model is a variational autoencoder with the theory behind it explained in Section 2.6

5.3.1 Architecture

The architecture is defined according to the AutoencoderKL class of the MONAI framework. As explained earlier, the architecture consists of two models.

The first model is called the encoder and consists of downsampling the input into a latent size which is distributed as a Gaussian centered to 0 with a standard deviation equal to 1. This model is composed of convolutional layers with 3D stride of size 2. The spatial dimensions are divided by 2 after each convolutional layer. Alongside these convolutional layers, there is ResBlock such as defined in 2.3 and self-attention block 2.7 which has proven to improve the accuracy of autoencoder models. The second model, the decoder, reconstructs the image in the initial domain from its latent code. The model is symmetrical to the encoder and composed of similar blocks. Two tables with exhaustive details about the architecture are available in the appendix A.

MONAI provides different parameters for their AutoencoderKL class:

- **spatial_dims**: The spatial dims refer to the number of dimensions in the original input.
- **num_channels**: The number of channels is a sequence of block output channels.
- **latent_channels**: The latent channels correspond to the number of features in the latent space.
- **num_res_blocks**: The number of ResBlock represents the number of residual blocks per level. These ResBlocks help to converge and mitigate the Vanishing Gradient Problem.
- **norm_num_groups**: The number of groups for the GroupNorm layers. GroupNorm normalizes the input across groups of channels, rather than across the entire batch or individual channels.
- **attention_levels**: The attention level is a sequence of levels to add attention.

Except for the spatial dims equal to 3 which refer to the dimensions of the original CTs (x,y,z), the other parameters are defined based on a trade-off between model complexity and the quality of the reconstruction obtained. The parameter **num_channels** influences the complexity of the model. Too few would prevent the model from correctly learning the patterns/structures of the scans and too many would make the model more complex. The parameter **latent_channels** also comes from a trade-off between the reconstruction quality of the VAE and the complexity of the latent space. Too many features in the latent space would make the diffusion model more complex.

To find the best possible configuration, a grid search was carried out for the parameters highlighted above. Here is the best-performing configuration found during the experiments:

- spatial_dims=3,
- num_channels=(64, 64, 64),
- latent_channels=3,
- num_res_blocks=2,
- norm_num_groups=32,
- attention_levels=(False, False, True)

5.3.2 Training

Losses

The loss used during the training process is a combination of different losses as presented in [44]. This combination is composed of:

- L1Loss
- KlLoss
- PerceptualLoss
- PatchAdversialLoss

The L1Loss [3] is the classical mean absolute error between each voxel from the original CT and the reconstructed CT, we call it the reconstruction loss:

$$\ell(x, y) = L = \{l_1 \dots l_N\}^T, \quad l_n = |x_n - y_n|$$

[3] https://pytorch.org/docs/stable/generated/torch.nn.L1Loss.html#torch.nn.L1Loss

The KlLoss is the Kullback-Leibler (KL) divergence between the learned distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and a standard Gaussian distribution. The μ and σ are estimates during the training. It helps to structure the latent space of the model, it is described in greater detail in 2.6.

The third loss is the perceptual loss [4], which compares the similarities between the reconstruction output of the network and the original CT. This metric comes from [57], in which the authors point out that this metric measures similarity in a way that aligns with human judgment while most traditional metrics assume pixel-wise independence. Perceptual loss uses features from pre-trained deep neural networks. The squeeze model with the is_fake_3d parameter set to True is used. This parameter is used to calculate the 2D perceptual on slices from the three axis instead of the 3D image.

Lastly, there's the PatchAdversialLoss which takes into account a patch discriminator. This discriminator is a Patch-GAN discriminator based on Pix2PixHD [52]. Instead of classifying the entire image as real or fake, the Patch-GAN discriminator operates on smaller patches of the image. This discriminator and the loss are used to learn the structures and details of the image more accurately and have proved their effectiveness in many applications [24].

The overall cost function is a weighted sum of these different loss functions:

$$\begin{aligned} Loss = & L1Loss + 10^{-6} * KlLoss + 10^{-3} * PerceptualLoss \\ & + 10^{-2} * PatchAdversialLoss \end{aligned}$$

The different coefficients are used to limit the impact of different losses. These coefficients are based on those pre-selected in the MONAI framework tutorials. The weight for the KlLoss is the lowest because the value is very large so the very low coefficient is used to limit the impact. The coefficient of the PerceptualLoss is lower than the PatchAdversialLoss's coefficient since the PerceptualLoss is computed as a sum of 2D PerceptualLoss. As the reconstruction loss is the most interesting, it is not attenuated. However, as it has been shown by [44], these different losses have a real impact on the quality of the reconstruction of the variational autoencoder and its convergence.

[4] `https://docs.monai.io/en/stable/losses.html#perceptualloss`

Optimizer

The optimization method used during all the training experiments is the Adam (Adaptive Moment Estimation) optimizer introduced by Kingma et al [28]. This algorithm is inspired by other stochastic optimization methods such as AdaGrad or RMSProp which are based on gradient descent. Adam adjusts the learning rate for each parameter individually, which allows it to adapt to the learning dynamics of different parameters. Adam incorporates momentum, which helps accelerate gradient vectors in the right directions, leading to faster converging. According to Kingma et al [28] the method is "computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters". This optimizer is one of the most widely used in the world of Deep Learning.

5.4 Diffusion Model

This section describes the second model of the pipeline, its architecture, and the training process. This model is a diffusion model with the theory behind it explained in Chapter 3

5.4.1 Architecture

The architecture is defined according to the DiffusionModelUNet class of the MONAI framework. The principle is to use a U-net architecture to identify the noise added to an image. The encoder-decoder structure of U-Net is well-suited for the denoising task in diffusion models. The encoder can capture complex features from the noisy input, and the decoder can reconstruct the clean image from these features. U-Net's skip connections allow the model to utilize both local (high-resolution) and global (low-resolution) features effectively, which is crucial for accurate denoising and generating high-quality images.

The MONAI framework adds a few specific features to the architecture presented in Section 2.4. For example, the U-net is conditioned based on a timestep to represent the different timesteps of the diffusion model. Another specific feature is the use of the SiLU[5] activation function instead of the ReLU[6] in the convolution layers. This activation function is smooth and differentiable everywhere. Unlike ReLU, which is a piecewise linear and monotonic function, SiLU is non-monotonic. This means SiLU can capture more complex patterns and dependencies in the data, which might lead to

[5] `https://pytorch.org/docs/stable/generated/torch.nn.SiLU.html#silu`

[6] `https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html#relu`

better feature extraction and more effective denoising. Finally, SiLU provides a non-zero gradient for both positive and negative inputs, avoiding the "dying ReLU" problem where neurons can become inactive and stop learning if they get stuck in the negative side of the ReLU. This consistent gradient flow can improve the training dynamics, especially in deeper networks. A table detailing each layer of the architecture is available in the appendix B.

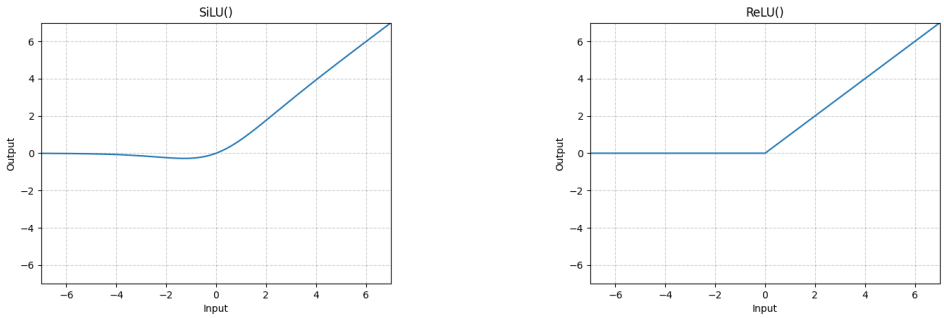

Figure 5.2: Graphical comparison between SiLU and ReLU. Credits: [6]

In addition to the skip connection that propagates low-level information between the encoder and the decoder, there is an attention block in between each downsampling/upsampling operation. By adding these layers and incorporating attention mechanisms, the network could better capture long-distance dependencies and contextual relationships between voxels at different scales.

MONAI framework provides two more parameters than VAE. The first is the number of channels in the model's input and output. The second is the number of channels in each attention head.

Here is the best-performing configuration found during the experiments:

- spatial_dims=3,
- in_channels=3,
- out_channels=3,
- num_res_blocks=2,
- num_channels=(64, 64, 64),
- attention_levels=(False, True, True),
- num_head_channels=(0, 64, 64)

5.4.2 Training

There are several components involved in training a diffusion model. The first is the scheduler. The MONAI-powered scheduler is based on [22]. The scheduler is used to control the noise added at each timestep. The scheduler is defined based on the number of timesteps in the Markov chain and the noise addition method (e.g., constant addition regardless of the timestep or linear interpolation to add less noise at the start of the chain). The literature mentioned above recommends the following configuration:

- Number of timesteps = 1000
- Method = scaled linear beta
- Beta start = 0.0015
- Beta end = 0.0195

This noise scheduler can be seen visually by taking the cumulative function of $\alpha_t = 1 - \beta_t$. The figure below represents this cumulative function for the recommended configuration.

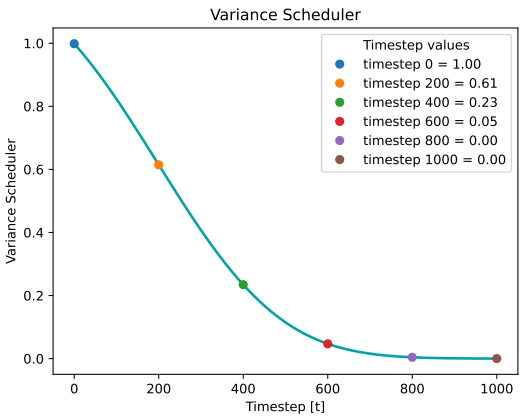

Figure 5.3: Visualization of the noise scheduler for the forward pass

The model is trained using the same optimizer as the VAE and as follows. First, a noise sample is randomly generated from a Gaussian $\mathcal{N}(0, 1)$ in the latent space. Then, a timestep t between 0 and 1000 is also randomly generated from the Uniform distribution. Based on these two random parameters, the input at timestep t to the forward process is generated x_t and therefore the real added noise can be recovered. U-net aims to predict this added noise based on the input x_t and the timestep t. Training is then carried out by minimizing the error between the predicted noise and the true noise with MSE.

5.5 Inference

The MONAI framework has implemented the inference process the same way as described in the original paper [22].

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **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 5.4: Sampling algorithm of the diffusion model. Credits: [22]

The first step of the algorithm is to initialize a random vector from a Gaussian $\mathcal{N}(0, 1)$ of the size of the CT in the reduced space. The aim is then to refine the random sample using the reverse process iteratively. When this process is complete, the $\mathbf{x}_0$ sample is decoded with the decoder part of the VAE trained earlier. Figure 5.4 represents the pseudocode of this process.

During the sampling operation, retrieving the intermediary steps that produce the final synthetic CT is possible. Figure 5.5 is an example of such a process. With these seven intermediate steps, it is visible that the model gradually denoises the 2d slices 1, 32, 64, 96, and 120 of the 3d synthetic CT. Slowly the model starting from complete random Gaussian noise removes the noise toward a CT close to the distribution of the dataset. This is the visualization of the for loop in the Sampling algorithm pseudo-code.

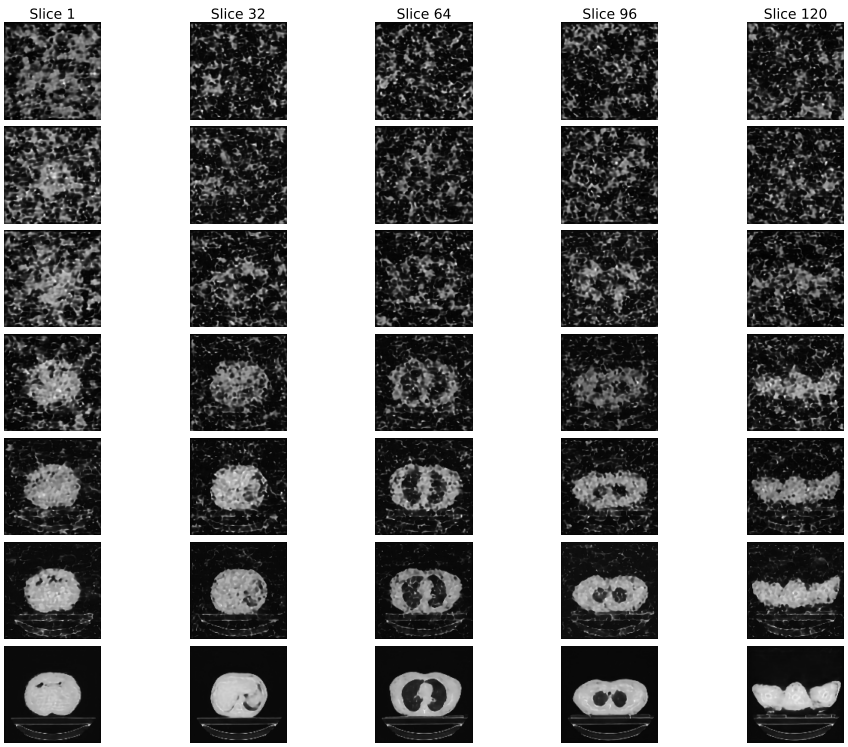

Figure 5.5: Visualization of the reverse process during the sampling operation

Chapter 6

Results and Interpretations

This chapter presents a comprehensive analysis of the experiments conducted to evaluate the performance of the models. This section is divided into two parts, the first dealing with VAE and the second with the diffusion model. Various evaluation metrics, such as Structural Similarity Index (SSIM), Mean Squared Error (MSE), and t-SNE visualization of the latent space, are employed to assess the quality of the reconstructions and the organization of the latent space. Detailed results and insightful visualizations are provided to demonstrate the efficacy and limitations of the model, offering a clear understanding of its strengths and areas for improvement.

6.1 VAE

Multiple configurations of the model were trained and analyzed to thoroughly evaluate the performance of the VAE. This subsection focuses on a comparative analysis of two specific configurations. The first is presented in Section 5.3 and uses 3 latent channels while the second uses 128 latent channels. Both configurations were trained on Nvidia Tesla A100 80G GPUs to ensure consistent training conditions and to leverage high computational power for efficient model optimization. Moreover, both configurations have been trained for 48 hours which is the maximum allocated time possible. The comparison aims to highlight the impact of latent space dimensionality on the quality of image reconstructions and the overall performance of the VAE.

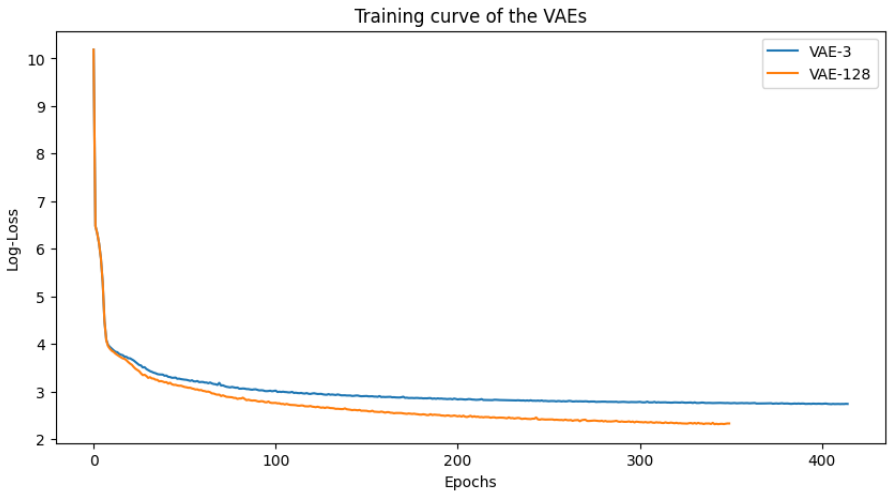

Figure 6.1: Training Curve of the VAEs

Figure 6.1 shows the evolution of the loss function, introduced in Section 5.3, during model training. The model with 128 latent channels demonstrates a faster convergence rate and achieves a smaller loss value compared to the model with 3 latent channels. However, the model with 128 latent channels executes fewer epochs (-65) within the same time. This difference in performance is attributed to the number of parameters in each model: the model with 128 latent channels has 4,584,193 parameters, while the model with 3 latent channels has 4,102,568 parameters, representing an 8.5% increase in the number of parameters. The larger latent space allows for a better representation and learning of the underlying structures in the images, leading to improved performance.

6.1.1 Quantitative evaluation

Quantitative evaluation is a crucial aspect of model assessment that involves the use of numerical metrics to objectively measure and compare the performance of different models. It provides a clear and unbiased way to assess the accuracy, efficiency, and robustness of the models, ensuring that their performance is not solely based on subjective visual inspection.

To compare these two VAEs, 5 metrics are used, 3 of which are presented in previous Sections 2.1 5.3: Mean Square Error, Perceptual Loss, KL divergence. The last two metrics are used to measure image compression quality.

The peak-signal-to-noise ratio (PSNR) is a widely used metric for evaluating the quality of reconstructed images, particularly in the context of compression and noise reduction algorithms.

$$PSNR = 10 \cdot \log_{10} \left(\frac{d^2}{MSE} \right) \tag{6.1}$$

where d is the maximum possible value for a pixel. The PSNR value approaches infinity as the MSE value approaches zero, showing that a higher PSNR value implies better image quality. At the other end, a low PSNR value means large differences between images.

The other metric used is the Structural Similarity Index (SSIM) [54]. This metric is another perceptual metric that measures the similarity between two images. It is designed to take into account changes in structural information, luminance, and contrast. SSIM provides a more accurate assessment of image quality by considering human visual perception. An SSIM value of 1 means perfect similarity between the compared images, a value of 0 indicates no similarity and a value of -1 indicates perfect dissimilarity.

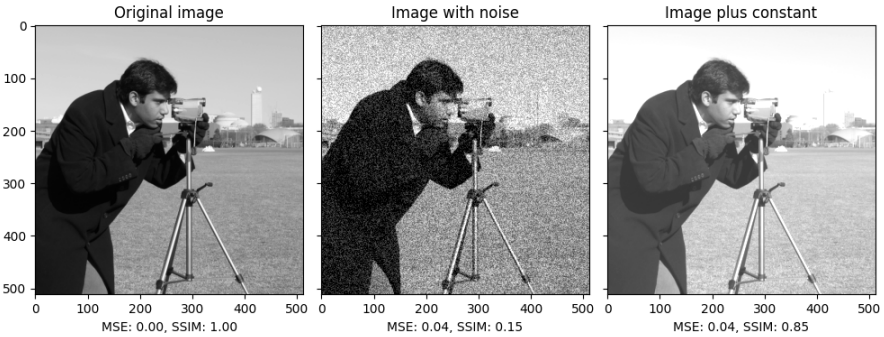

Figure 6.2: Comparaison between MSE and SSIM. Credits: [7]

Figure 6.2 shows that two reconstructions with the same MSE value can have large differences in terms of visual perception. The results of the quantitative assessments based on the 5 metrics are as follows:

Metric	**VAE-3**	**VAE-128**
MSE	1509.2502	484.5116
Perceptual Loss	0.0732	0.0188
KL Divergence	331643.90	1116400.75
PSNR	28.3214	33.3278
SSIM	0.9491	0.9810

Table 6.1: Metrics for Training set.

Metric	VAE-3	VAE-128
MSE	1697.2557	513.3570
Perceptual Loss	0.0788	0.0193
KL Divergence	335485.72	1146285.62
PSNR	27.7761	33.0603
SSIM	0.9426	0.9793

Table 6.2: Metrics for Testing set.

The data presented in Tables 6.1 and 6.2 demonstrate that the VAE with 128 latent channels reconstructs images more accurately than the VAE with 3 latent channels. Specifically, for both the training and testing sets, the VAE-128 outperforms the VAE-3 in terms of pixel-level metrics, such as MSE and PSNR. Furthermore, the higher Perceptual Loss observed for the VAE-3 indicates that its reconstructions are less perceptually similar to the original images, while the SSIM values, which are closer to 1 for the VAE-128, further confirm its superior reconstruction quality.

However, the KL divergence metric reveals that the VAE-3 exhibits a more structured latent space, as its distribution is closer to the standard Gaussian. This observation is crucial because, as discussed in Section 2.6, the primary goal of a VAE is to learn representations in a structured latent space that can faithfully reconstruct the original images. The VAE-128 achieves more faithful image reconstructions due to the increased complexity of its latent space, although this same complexity presents challenges in maintaining a structured latent space.

6.1.2 Qualitative evaluation

Qualitative evaluation is an essential component of model assessment, complementing quantitative metrics by providing a visual and perceptual understanding of the model's performance. Qualitative evaluation involves the direct examination of reconstructed images to assess their visual quality, realism, and adherence to the original content.

Figure 6.3 confirms the conclusions about the quality of the reconstruction obtained with the quantitative assessment. The VAE-128 provides a greater level of detail. However, the VAE-3 still manages to represent the structures present in the initial scan.

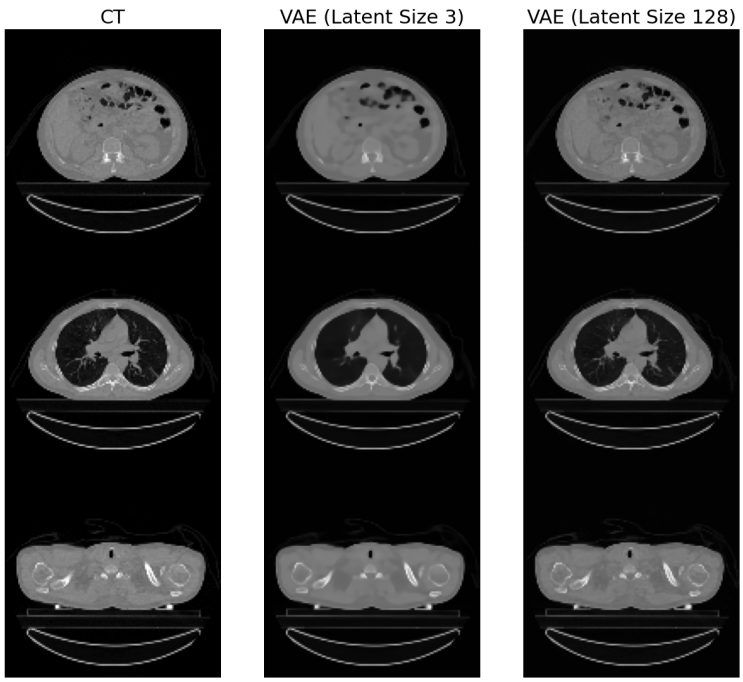

Figure 6.3: Visual comparison of the VAEs

Figure 6.4 illustrates an example of reconstruction errors for an independent slice. Both models encounter challenges in accurately representing the intricate details within the lungs. This difficulty may arise from the high variance in voxel values between the air-filled regions (-1000 to -100) and the surrounding tissues. The primary distinction between the models lies in the magnitude of these reconstruction errors, with noticeable differences in error intensity.

Figure 6.4: Visual comparison of the absolute errors.

6.1.3 Latent Space Comparison

A final assessment is the evaluation of the latent space of the various VAEs. The initial step involves visualizing the various latent spaces to identify potential differences. To achieve this, a clustering algorithm was applied to the 128x128x128 scans, which were flattened into a 1D array. Specifically, the KMeans++ [3] algorithm with 5 clusters from the sklearn library[1] was used. The precise configuration of the KMeans++ algorithm (other meta-parameters) is not crucial, as the goal is to compare the clusters' positions within the latent spaces.

In order to visualize the latent space, a dimension reduction algorithm is applied to the latent representations of the scans. The algorithm used is a t-SNE. t-distributed stochastic neighbor embedding (t-SNE) from [50] is a non-linear dimensionality reduction technique that is a variation of Stochastic Neighbor Embedding (SNE). The principle of t-SNE relies on minimizing a KL divergence between the similarities of all pairs of points in a dataset of N dimensions. It is then possible to visualize complex data such as genes, in a 2D or 3D space. This t-SNE is applied to the latent flattened representations of VAEs. The Python implementation used for the t-SNE relies on sklearn library[2].

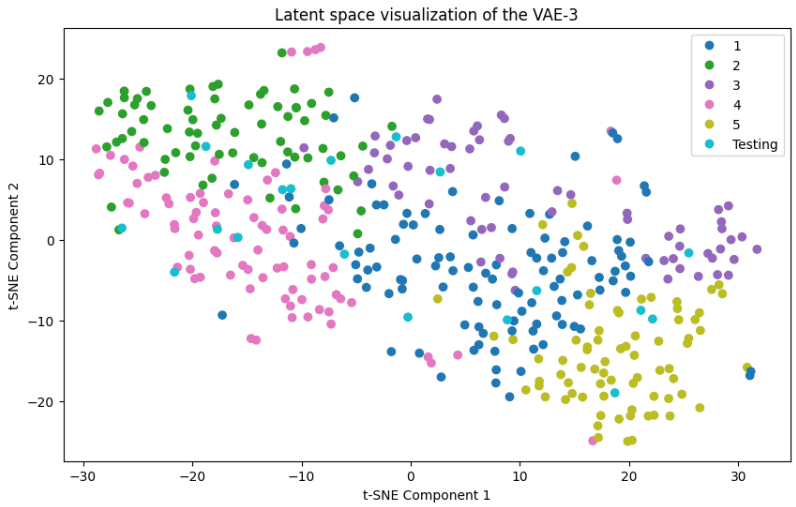

Figure 6.5: Visualization of the VAE-3 latent space

Figures 6.5 and 6.6 highlight two important observations. Firstly, in both latent spaces, the test images are well-distributed, indicating that the latent spaces are well-structured and generalize effectively to the CT scans in the testing set. Secondly, some of the clusters found in the initial space are clearly delimited in the visualizations. However, the clusters appear more

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html`

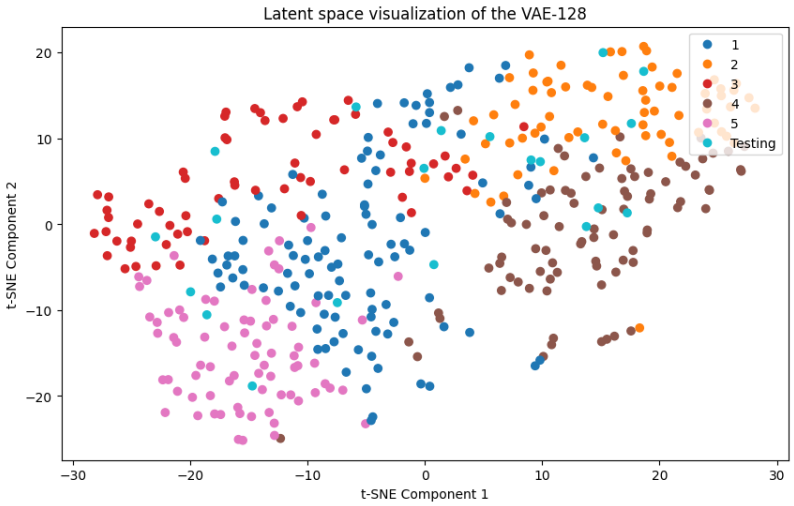

Figure 6.6: Visualization of the VAE-128 latent space

spread out in the latent space visualization of VAE-128. This suggests that VAE-3 has better preserved the data structure in its latent space, supporting the interpretation from the KL-divergence evaluation. It is also important to note that the latent space of VAE-128 has significantly more dimensions than that of VAE-3, which may contribute to this difference.

A final analysis of the latent space of VAEs involves evaluating the smoothness and continuity of the space. This assessment typically includes performing linear interpolation between points in the latent space and decoding these points into images. Figure 6.7 illustrates the comparison of interpolation results between randomly selected scans. The findings demonstrate a similar level of smoothness and continuity in both latent spaces. This consistency suggests that the spaces maintain structural integrity. In contrast, if the spaces were not continuous, the interpolation would yield erratic, noisy images lacking meaningful transitions.

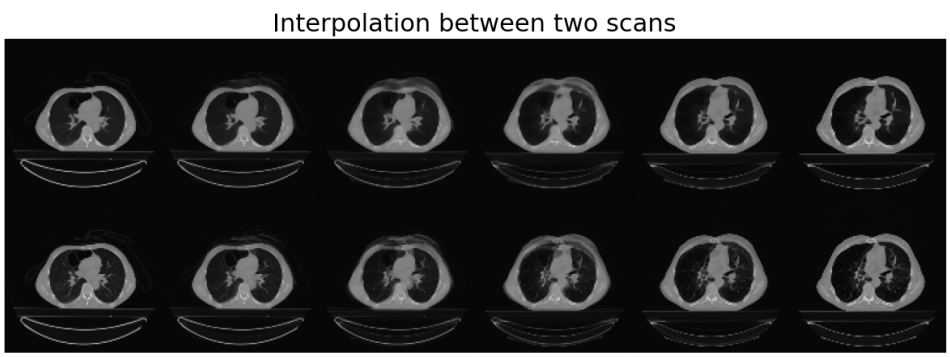

Figure 6.7: Interpolation between 2 scans latent space. The first row is VAE-3 and the second is VAE-128

6.2 Diffusion

The diffusion models are trained in the latent space of a VAE, meaning the choice of VAE significantly impacts the diffusion process. Initially, various diffusion models were trained on the two VAEs compared in the previous section. However, no combination of the VAE-128 and a U-net configuration yielded convincing results. This lack of success is attributed to the excessively large latent space of the VAE-128. Figure 6.8 represents a generated sample from the optimal combination of VAE-128 and U-net configuration found. As a result, only VAE-3 was selected for further testing.

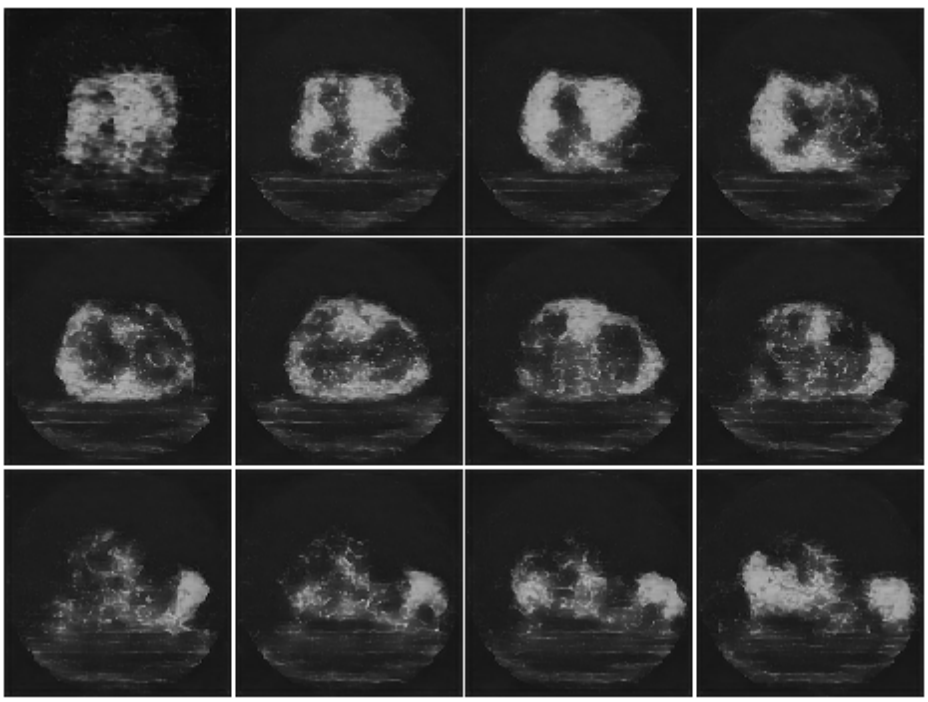

Figure 6.8: Sample generated using the optimal combination of VAE-128 and U-net configuration.

The best diffusion model obtained during experiences was by training the U-net with parameters defined in 5.4, with the VAE-3 model presented in the previous section for the dimensionality reduction of the inputs. The model was trained for a number of 1500 epochs, it is calculated so that the training time is approximately 48 hours, which is the limit of time it was possible to train the model on a Nvidia Tesla A100 80G GPU. As a reminder, when training a diffusion model the loss is computed on noise prediction and not on the comparison between the sampled CT and the CT from the dataset. It is always interesting to see how this noise develops during training:

As expected for any kind of deep learning model, at the start of the training, the network has trouble making good predictions for the amount of noise present in the CT. However, it is possible to observe a net progression of

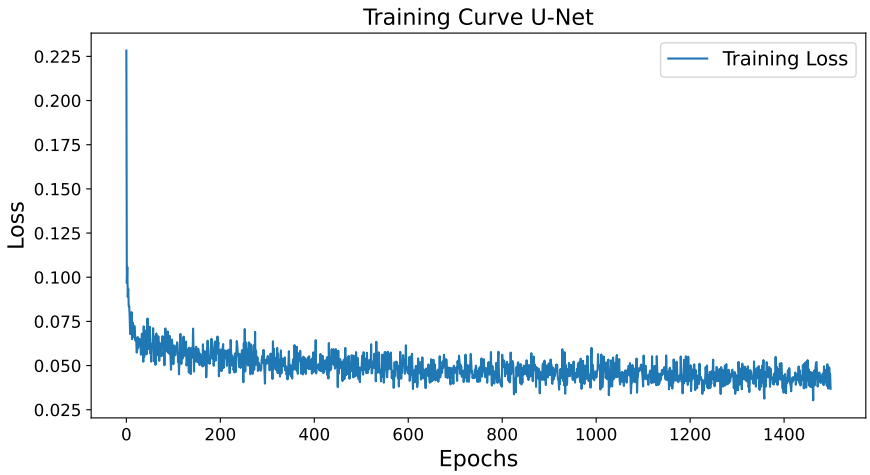

Figure 6.9: Training Curve of the U-net model

the mean squared error loss during the first 400 epochs of the training. While the loss does not improve afterward, the quality of the synthetic samples continues to improve. Indeed, more training would probably increase the quality of the samples, as is the case for many published projects such as GenerateCT [19] or Stable Diffusion [44] who train their models for several weeks on multiple GPUs.

6.2.1 Quantitative evaluation

The quantitative evaluation of the diffusion model has the same goal as for the VAE. However, it is separated into two parts, firstly an assessment of the generated sample's quality compared to the original dataset. Secondly, the diversity of the synthetic samples is evaluated.

Sample Quality

The Fréchet inception distance (FID) is the most commonly used metric to assess the realism and quality of synthetic images inferred by GANs or Diffusion models, the smaller the metric the better the quality. To compute the metric, the real features of the image are compared with the synthetic features of the generated image. For this purpose, a VAE-16 has been trained on the dataset to extract the necessary image features. The latent dimensions are then 16x32x32x32 which would make about 524,288 features to compare. This number is too high; therefore, the FID is computed at the 2D slice level. The autoencoder extracts about 16,384 features, which is more reasonable but still high compared to the 2,048 features extracted by the InceptionV3 model [49], which is frequently used to extract features to compute FID scores.

Batch Size	FID
50	5299.9884
50 (filtered)	4785.7112
100	7462.5603
100 (filtered)	7256.4809

Table 6.3: FID scores for different numbers of synthetic samples either random or pre-selected.

The scores presented in Table 6.3 were calculated using two different CT batches. The first batch of 100 CT scans was manually filtered from 500 samples to ensure enough quality inferences. The manual evaluation involved visually analyzing each of the 500 samples individually. Samples without noise were selected as correct. This previous step was performed because the model might produce poor generations that could negatively influence the metric. An example of this can be found in the Section 6.2.2. Indeed this hypothesis can be verified directly by comparing the results for 50 random samples versus the 50 pre-selected ones. There is a rough difference of 500 FID scores between the two.

Sample Diversity

In this part of the evaluation, the diversity of generated samples is evaluated. The assessment of this diversity is done via two metrics. The first one is the SSIM as defined in 6.1.1. The second one is an extension of the SSIM called Multi-Scale Structural Similarity Index (MS-SSIM), as the name indicates, it computes SSIM on different scales to capture information at different levels of detail. The metrics are calculated between synthetic images to demonstrate the diversity among them.

Batch Size	SSIM	MS-SSIM
50	0.1472	0.2576
50 (filtered)	0.1660	0.3168
100	0.1417	0.2622
100 (filtered)	0.1628	0.3132

Table 6.4: Diversity metrics applied on synthetic samples either random or pre-selected.

The metrics presented in Table 6.4 have been computed on different batches for the same reason as in Section 6.2.1. The results show a general diversity between the synthetic samples since the metrics are close to 0, and 0 SSIM or MS-SSIM would mean perfect diversity. Both metrics support the decision to use pre-selected synthetic CTs, as the non-filtered CTs are more diverse, likely because some of the model's generations are close to random noise, or generate a lot of air (values between -1000 and -100 on the Hounsfield scale).

6.2.2 Qualitative evaluation

High Quality sample

In this initial stage, CT images are visually assessed and considered correct if they meet reasonable quality standards. The figure below displays three different synthetic CTs with 2D slices taken from five distinct regions.

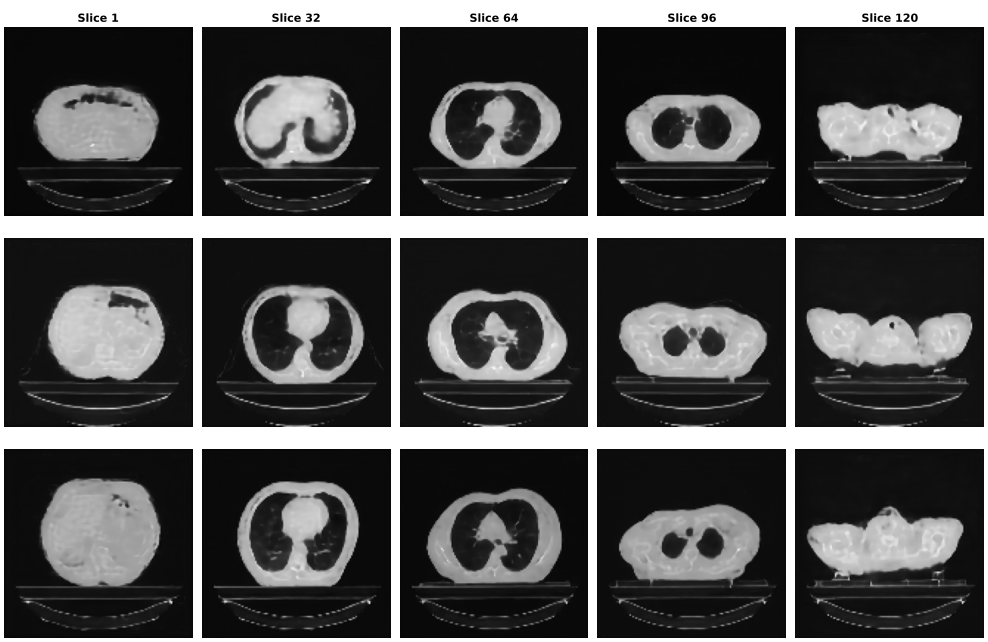

Figure 6.10: High-quality sample from the diffusion model.

The key takeaway from Figure 6.10 is that for most of the generated samples, the model can produce CT scans that capture the general features of the original dataset, including the body contours, lungs, detailed structures within the lungs, and shoulders.

Poor Quality sample

The visual assessment of poor-quality synthetic samples is more evident and straightforward. Below are four synthetic CTs that represent most of the inference problems encountered by the model.

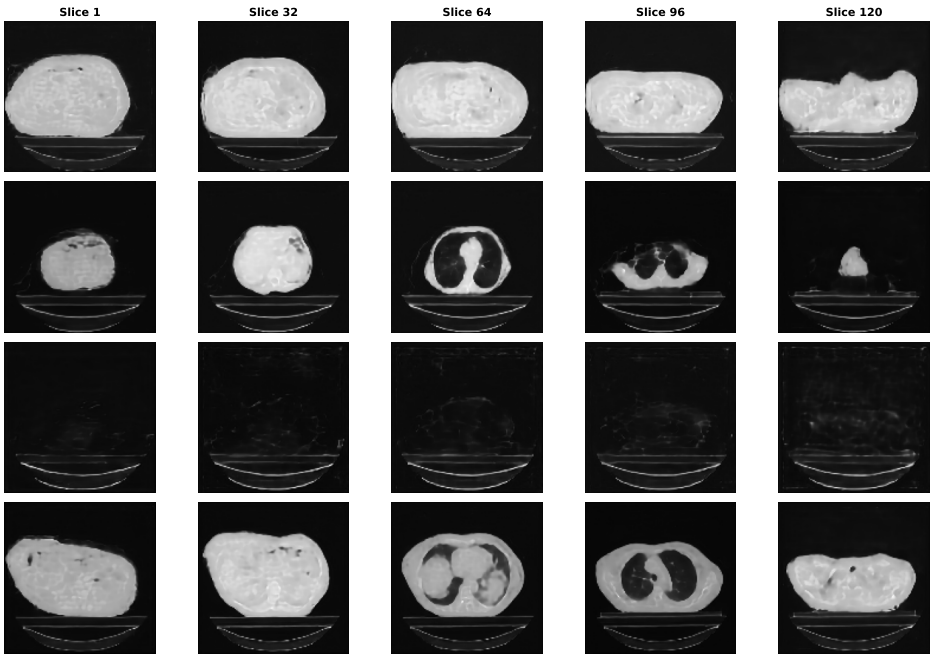

Figure 6.11: Poor-quality sample from the diffusion model.

- **First CT**: For all the slices, the model outputs a body that can be considered correct but has no lung inside of it.
- **Second CT**: The second CT is a mismatch of everything, there is no consistency between the shape of the body throughout the slices. At slice 96 there is no delimitation of the body anymore and the lung seems to be disappearing in the air. For the last slices, there are no more shoulders the model outputs only a portion of the body floating above the table.
- **Third CT**: Only the generation of the table is correct, the rest is almost only black which is the air in the Hounsfield unit.
- **Fourth CT**: Two problems are observable for this set of slices, firstly there is a distortion of the body on the upper left part. Secondly, the lungs appear late on the slices which would mean anatomically that the lungs are located in a higher area of the thoracic cavity than they should.

6.2.3 Latent Space Comparison

The aim of this experiment is to evaluate the distributions of the generated scans with those of the training scans. To do this, a dimension reduction was performed on the scans using a t-SNE which minimizes the KL divergence between the similarities of data in the original dataset versus synthetic images of the same size of the dataset.

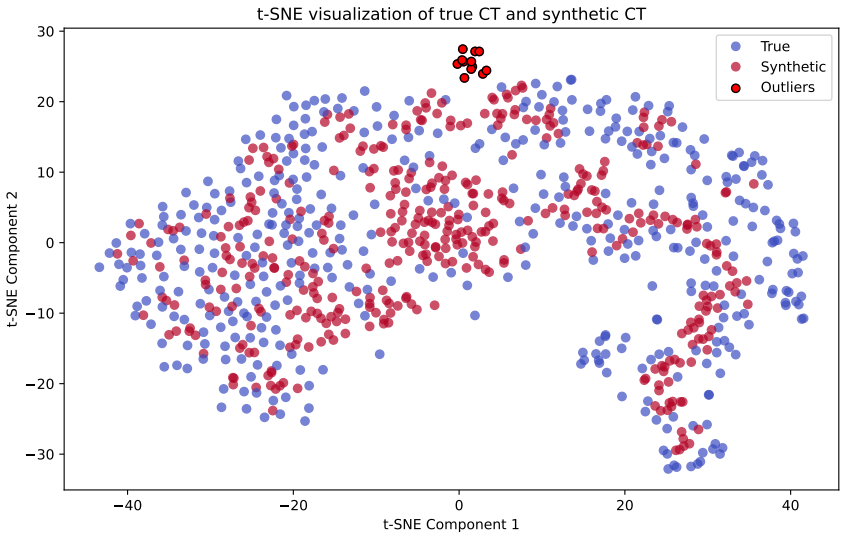

Figure 6.12: t-SNE Algorithm applied on synthetic versus real data

At first sight on the graphic above, it is possible to observe that many points from both classes are well mixed. Even more, by looking closely, many synthetic points are taking the same form as those from the true dataset. This could mean that the diffusion model captures the essence of the distribution from the true data. Also, at (0,25) on the (x,y) axis, an agglomeration of points could be defined as outliers. One hypothesis is that these outliers are bad samples close to random noise. Finally, since the algorithm is not deterministic, the same dataset may produce slightly different visualizations upon multiple runs, which is why multiple runs of the algorithms have been done to ensure the consistency of the results across them.

The verification of hypothesis mentioned above is verified by analyzing the points of synthetic data that produce the outliers, the twelve outliers can be viewed in the appendix C.

Chapter 7

Discussions

During this thesis, we identified several points that merit discussion to continue or improve this research project. In addition with all the results exposed so far, we would like to discuss more on some limitations we encountered during the thesis and on the workflow developed. In fact, this thesis involved several technical constraints for which we have identified alternatives to overcome them. The purpose of this chapter is to discuss the workflow developed in general and outline the problems and solutions identified to improve our results' quality.

7.1 Workflow

The workflow developed for the latent diffusion model in the context of generating CT scans of lungs with tumors has numerous potential applications in the medical field. This work could be used in the development of computer-aided diagnosis (CAD) models to assist radiologists in detecting and characterizing lung tumors with greater accuracy. Additionally, it could serve clinical research by providing realistic synthetic data for testing and validating new medical image analysis methods, thus reducing reliance on real data, which is often difficult to obtain.

To make this workflow more complete and ready for clinical practice, several aspects need further development. It is important to improve the resolution of the generated images to make them comparable to the high-quality CT scans used in hospitals. Another way of making this workflow more complete is to make it more general by taking into account different modalities, i.e. generating scans of any part of the human body. Finally, adding the possibility for users to customize the generations would make the workflow more complete. This customization would be based on user input in order to influence the location or size of the tumor, for example. Generation would then be conditioned on the basis of the user's needs.

7.2 Improvements

7.2.1 Image resolution

The first way of improving the quality of our generated images is based on the initial resolution of our images. As described in 5.1 all the experiments have been done with a resolution of 128x128x128 for the CT scans. However, before pre-processing, CT scans were made up of several 512x512 resolution slices. The number of slices differs from one scan to another, which is why we have standardized it at 128.

The main reason for compressing the scans is that the higher the resolution of the scans, the more GPU-intensive the VAE drive. Moreover, the VAE would be more complex and then the training would take more time. Consequently, the 128x128x128 resolution is a trade-off between the quality of the initial image after compression and the resources available for training the models. However, compressing images has an impact on the quality of the images generated. The generation process is driven by these compressed images, which have lost some of their quality. The VAE decoder enables the quality and level of detail of the initially compressed images to be maintained in the best of cases.

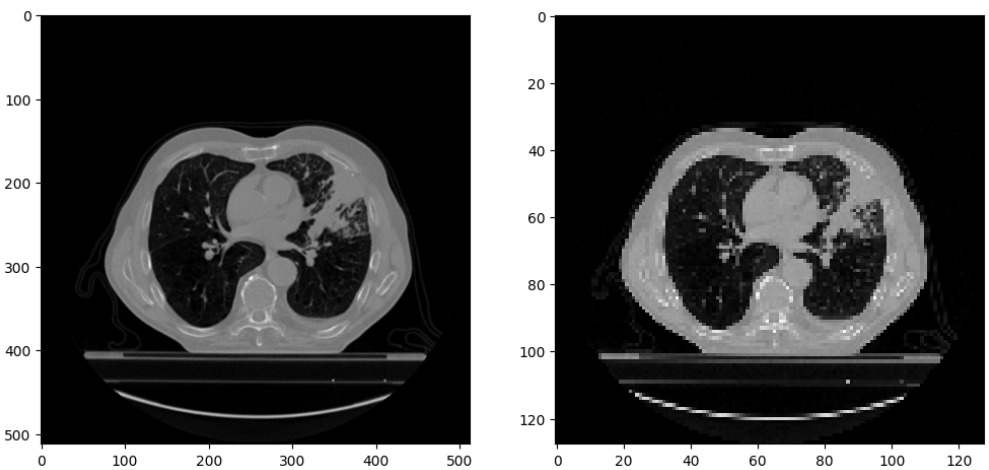

Figure 7.1: Impact of resolution compression

There are several possible solutions to compensate for the reduced level of detail shown in 7.1. The simplest solution would be to invest in more powerful GPUs to train the VAE without sharing resources. A second solution is to increase the resolution of the image in post-processing. This idea comes from the GenerateCT [19] paper, which uses a cascade model to upsample low-resolution slices from generated 3D chest CT volumes. The aim would be to add a third model that takes the image generated as input and returns the same image in a higher resolution with an increase in image quality.

7.2.2 Standardization of the dataset

A second way of improving the results of the models (VAE, Diffusion) is to apply more complex processing to the dataset scans. The original dataset used is a set of scans taken by different scanners with different configurations and in different years. These differences lead to a lot of variance between the scans.

Isolating the body

Firstly, it is interesting to extract the patient's body from the scan, allowing us to concentrate solely on the patient. For the moment, the models take into account the table on which the patient is positioned during the scan. Removing the table of scans allows the models to learn the patterns/structure of the body's anatomy only. In addition, the CT scans do not have the table at the same height, which leads to a variance in height between scans. If one body is higher than another, its lungs will also be higher, which makes learning anatomical structures more complex.

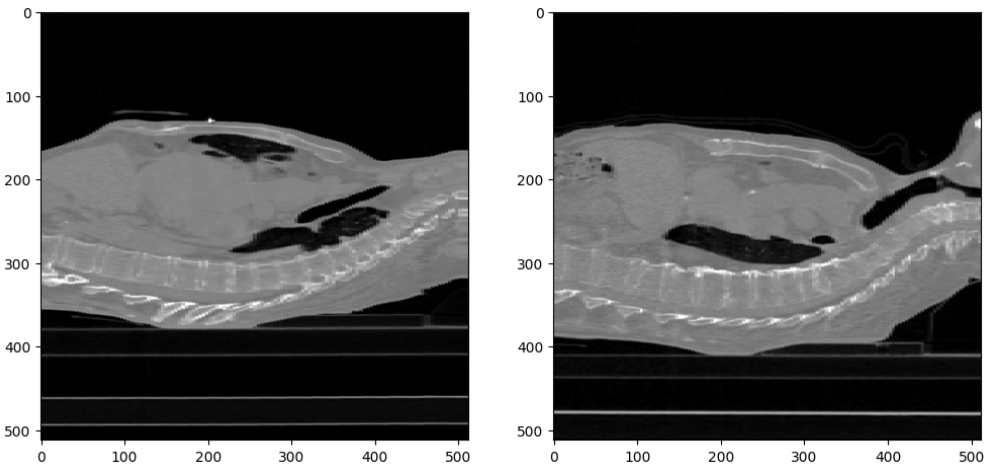

Figure 7.2: Height variance due to table

One solution for removing the tables from the scans is to identify the height of the table for each scan and reduce the intensity of the voxels below it to the minimum value, i.e. -1000. One way of identifying the height is to use the fact that the table is flat. However, we note in 7.2 that the scans include a small shelf below the shoulder blades, so the height of the table is not the same in the same scan. A second method is to use a segmentation tool. This segmentation tool would provide a mask of the whole body, which is missing from the original dataset. However, care must be taken because the intensity values corresponding to the table also correspond to those of certain tissues in the human body. It is also important to note that the intensity of the table is not constant in a CT and differs from the intensity of the table in other CTs.

Align bodies

Another difference identified between the scans is the area of the body considered. Some CTs start at the chin and end at the diaphragm, whereas others can start at the shoulders and end before the diaphragm. This difference causes misalignment between the CTs. For example, 7.3 shows that one patient's lungs end at the lower abdomen of another. This misalignment makes the learning task more complicated. A solution could be to crop the CTs in a supervised manner.

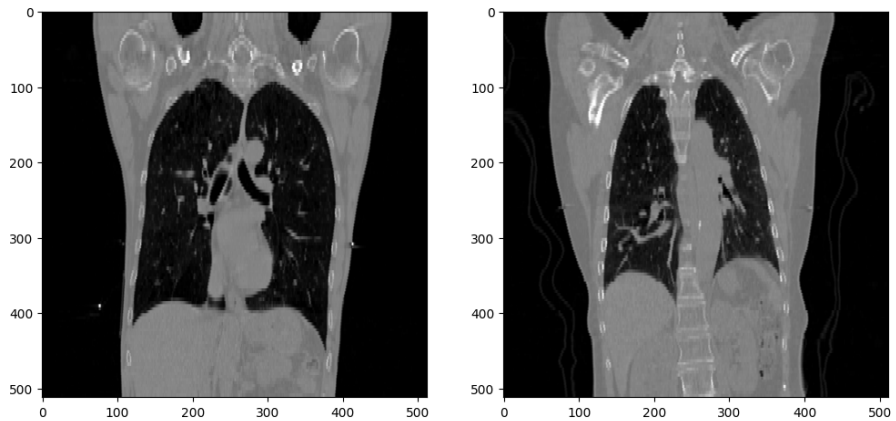

Figure 7.3: Illustration of wrong alignment

7.3 Conditioning

One way to continue the project is to add conditioning to the generation. During preprocessing, we extracted the mask corresponding to the main tumor of the scan (GTV-1) for each CT. The objective would be to use this mask to guide the generation of scans. The model would take a tumor mask as input and generate a scan with a tumor at the location of the mask.

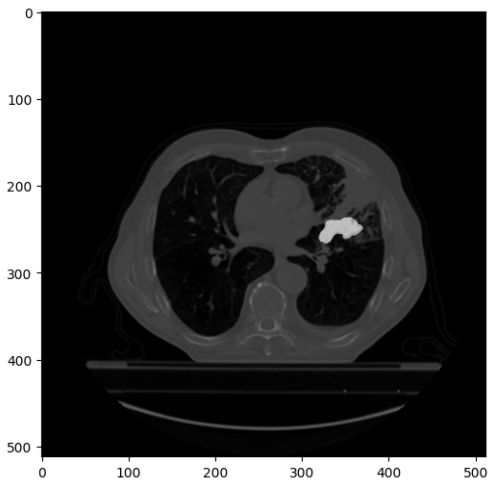

Figure 7.4: Illustration of CT with a mask

One way to achieve this conditioning is by concatenating the mask to the input of the U-net in the diffusion model. Zolnamar Dorjsembe et al [13]. implemented this approach by extending their previous work, 3D-DDPM [12], which adapted the vanilla DDPM model for generating 3D volumetric images. The diffusion model class used, implemented in the MONAI framework, already supports this conditioning via concatenation or through a cross-attention mechanism between the input and the conditioning variables [44] [40].

Chapter 8

Conclusion

This thesis set out to explore the generation of realistic 3D synthetic data in the medical domain. As it can be expected, acquiring such data in the real world is quite difficult even nowadays due to the cost required for the scan, the qualified personnel required, and all the other limitations that apply to handling personal data. Research into diffusion models is very advanced for 2D images. However, the only papers published on this subject for 3D CT scans involve very complex frameworks that require a lot of resources to train, such as GenerateCT [19]. The framework developed in this work is composed of two neural networks, one for reducing the complexity of the input data and the second for generating synthetic CTs.

The first block of the framework is a Variational Autoencoder. The model has been trained to reduce the size of the input, which was 128x128x128, by a factor of four. By doing this, the input complexity for the next model is reduced to 3x32x32x32. The first channel of size 3 represents the latent channels in which the VAE can capture more details of the features in the input data such as the edges, contours,... The result of the training is convincing, and the reconstruction of the images looks good in a qualitative way. The reconstruction loss is far from perfect, however, it is not a surprise since the data is not scaled between 0 and 1 but on its original scale between -1000,1000 Hounsfield units.

The second block of the framework is a Diffusion Model. The model has been trained on the latent representation of the encoder from the first block. By doing so, it reduces the complexity of the training process, and thanks to the decoder of the first block it outputs synthetic CT scans in the original dimensions of 128x128x128. However, even with this reduction of complexity, the results are not perfect. Looking at the quantitative results, the FID score shows poor inference quality. While a qualitative evaluation shows synthetic scans of correct quality. An improvement of the metric was made possible by evaluating a batch of pre-selected CT to ensure a minimum quality of inference. The results of the quantitative evaluation

of the diversity metrics show that there is diversity in the generation of synthetic images, which supports the choice of a diffusion network rather than a GAN.

Once both models of the framework are trained, it is possible to use them for inference of synthetic CT. Since the Diffusion model has been trained on low resolution, the inference time is relatively low. On a Nvidia TeslaA100 GPU, it takes about 10 seconds to make the inference of one synthetic CT scan. The framework still requires a powerful GPU to run since it has to instantiate both the models and their weights in memory.

From the evaluations of the models, there is still room for improvement. These could be due to the initial size of the images given to the VAE since 128x128x128 is still a low resolution compared to the standard slice definition of 512x512. Another area for improvement is in the pre-processing related to the content of the CT scans, by removing the table using a body mask for example. These improvements could also be made by adding conditioning during the diffusion process with the different masks available inside the dataset.

To conclude, this framework is the first step towards a more complete one that could handle higher resolution of CT scans and conditioning. While the current framework does not provide CT scans of sufficient quality, it is possible to improve it with all future works and improvement paths proposed. So that one day it might be used for academic purposes or to help radiologists or any medical expert in their work.

Appendix A

VAE

A.1 Encoder

Layer	**Operation**	**Kernel Size/Stride**	**Feature Map Size**
Input	-	-	1 x 128 x 128 x 128
Conv_in	Conv3d	3x3x3 / 1x1x1	64 x 128 x 128 x 128
ResBlock1	Conv3d + ResBlock	3x3x3 / 1x1x1	64 x 128 x 128 x 128
ResBlock2	Conv3d + ResBlock	3x3x3 / 1x1x1	64 x 128 x 128 x 128
Downsample1	Conv3d	3x3x3 / 2x2x2	64 x 64 x 64 x 64
ResBlock3	Conv3d + ResBlock	3x3x3 / 1x1x1	64 x 64 x 64 x 64
ResBlock4	Conv3d + ResBlock	3x3x3 / 1x1x1	64 x 64 x 64 x 64
Downsample2	Conv3d	3x3x3 / 2x2x2	64 x 32 x 32 x 32
ResBlock5	Conv3d + ResBlock	3x3x3 / 1x1x1	64 x 32 x 32 x 32
Attention1	AttentionBlock	-	64 x 32 x 32 x 32
ResBlock6	Conv3d + ResBlock	3x3x3 / 1x1x1	64 x 32 x 32 x 32
Attention2	AttentionBlock	-	64 x 32 x 32 x 32
ResBlock7	Conv3d + ResBlock	3x3x3 / 1x1x1	64 x 32 x 32 x 32
Attention3	AttentionBlock	-	64 x 32 x 32 x 32
ResBlock8	Conv3d + ResBlock	3x3x3 / 1x1x1	64 x 32 x 32 x 32
Norm	GroupNorm	-	64 x 32 x 32 x 32
Output	Conv3d	3x3x3 / 1x1x1	3 x 32 x 32 x 32

Table A.1: Encoder architecture of the VAE

A.2 Decoder

Layer	**Operation**	**Kernel Size/Stride**	**Feature Map Size**
Input	-	-	3 x 32 x 32 x 32
Conv_in	Conv3d	3x3x3 / 1x1x1	64 x 32 x 32 x 32
ResBlock1	Conv3d + ResBlock	3x3x3 / 1x1x1	64 x 32 x 32 x 32
Attention1	AttentionBlock	-	64 x 32 x 32 x 32
ResBlock2	Conv3d + ResBlock	3x3x3 / 1x1x1	64 x 32 x 32 x 32
ResBlock3	Conv3d + ResBlock	3x3x3 / 1x1x1	64 x 32 x 32 x 32
Attention2	AttentionBlock	-	64 x 32 x 32 x 32
ResBlock4	Conv3d + ResBlock	3x3x3 / 1x1x1	64 x 32 x 32 x 32
Attention3	AttentionBlock	-	64 x 32 x 32 x 32
Upsample1	Conv3d	3x3x3 / 1x1x1	64 x 64 x 64 x 64
ResBlock5	Conv3d + ResBlock	3x3x3 / 1x1x1	64 x 64 x 64 x 64
ResBlock6	Conv3d + ResBlock	3x3x3 / 1x1x1	64 x 64 x 64 x 64
Upsample2	Conv3d	3x3x3 / 1x1x1	64 x 128 x 128 x 128
ResBlock7	Conv3d + ResBlock	3x3x3 / 1x1x1	64 x 128 x 128 x 128
ResBlock8	Conv3d + ResBlock	3x3x3 / 1x1x1	64 x 128 x 128 x 128
Norm	GroupNorm	-	64 x 128 x 128 x 128
Output	Conv3d	3x3x3 / 1x1x1	1 x 128 x 128 x 128

Table A.2: Decoder architecture of the VAE

Appendix B

Diffusion

Layer	**Operation**	**Kernel Size/Stride**	**Feature Map Size**
Input	-	-	3 x 32 x 32 x 32
Conv_in	Conv3d	3x3x3 / 1x1x1	64 x 32 x 32 x 32
Time_embed	Linear + SiLU + Linear	-	-
DownBlock1	Conv3d + ResnetBlock	3x3x3 / 1x1x1	64 x 32 x 32 x 32
Downsample1	Conv3d	3x3x3 / 2x2x2	64 x 16 x 16 x 16
DownBlock2	Conv3d + ResnetBlock + AttentionBlock	3x3x3 / 1x1x1	64 x 16 x 16 x 16
Downsample2	Conv3d	3x3x3 / 2x2x2	64 x 8 x 8 x 8
DownBlock3	Conv3d + ResnetBlock + AttentionBlock	3x3x3 / 1x1x1	64 x 8 x 8 x 8
MiddleBlock	Conv3d + ResnetBlock + AttentionBlock	3x3x3 / 1x1x1	64 x 8 x 8 x 8
UpBlock1	Conv3d + ResnetBlock + AttentionBlock	3x3x3 / 1x1x1	64 x 8 x 8 x 8
Upsample1	Conv3d	3x3x3 / 2x2x2	64 x 16 x 16 x 16
UpBlock2	Conv3d + ResnetBlock + AttentionBlock	3x3x3 / 1x1x1	64 x 16 x 16 x 16
Upsample2	Conv3d	3x3x3 / 2x2x2	64 x 32 x 32 x 32
UpBlock3	Conv3d + ResnetBlock	3x3x3 / 1x1x1	64 x 32 x 32 x 32
Output	Conv3d + GroupNorm + SiLU	3x3x3 / 1x1x1	3 x 32 x 32 x 32

Table B.1: U-Net Architecture

Appendix C

Outliers t-SNE

This section of the appendix shows the outliers of synthetic CT scans found by the t-SNE algorithm. As expected the outliers are CT generated with a lot of noise or the air.

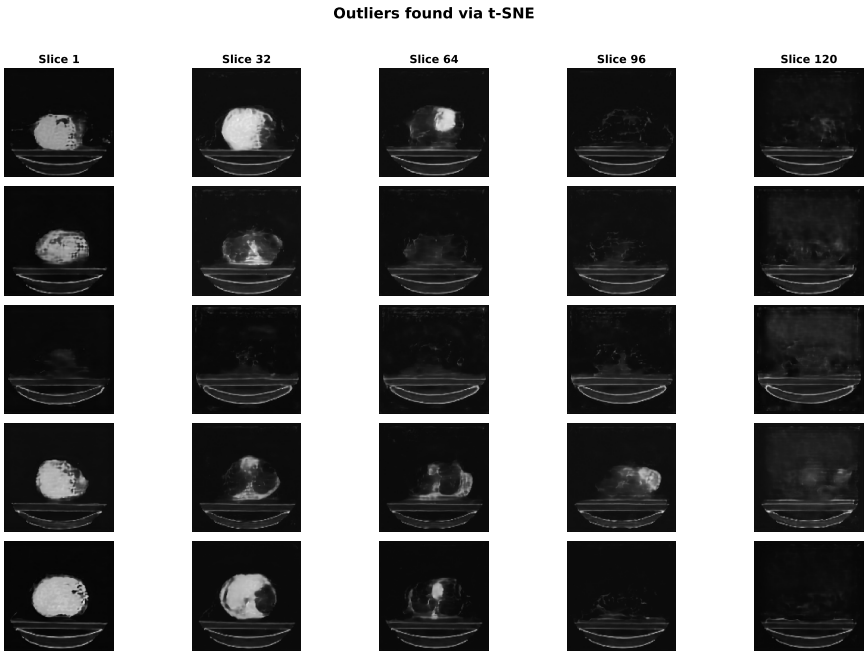

Figure C.1: Examples 1 to 5 of outliers synthetic CT found by t-SNE

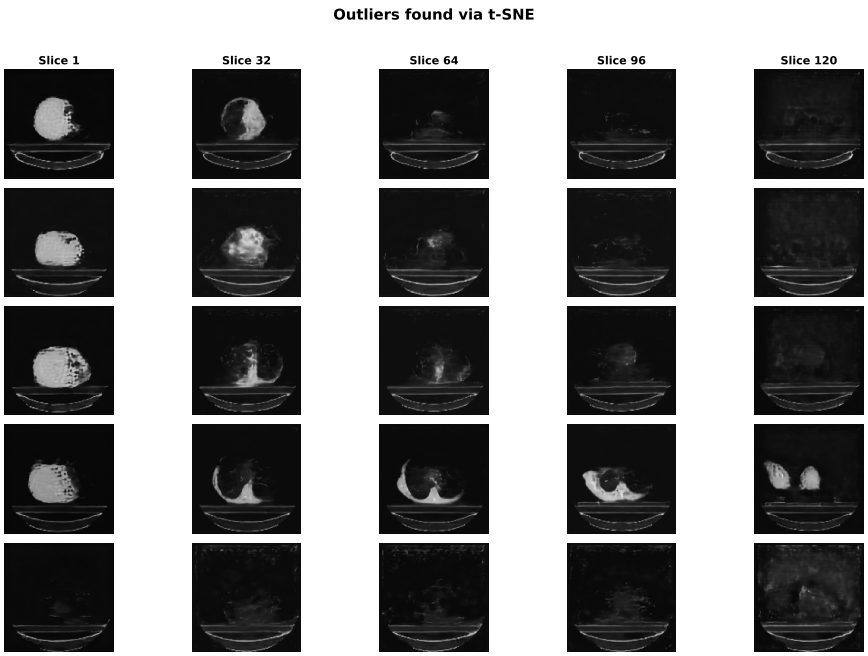

Figure C.2: Examples 6 to 11 of outliers synthetic CT found by t-SNE

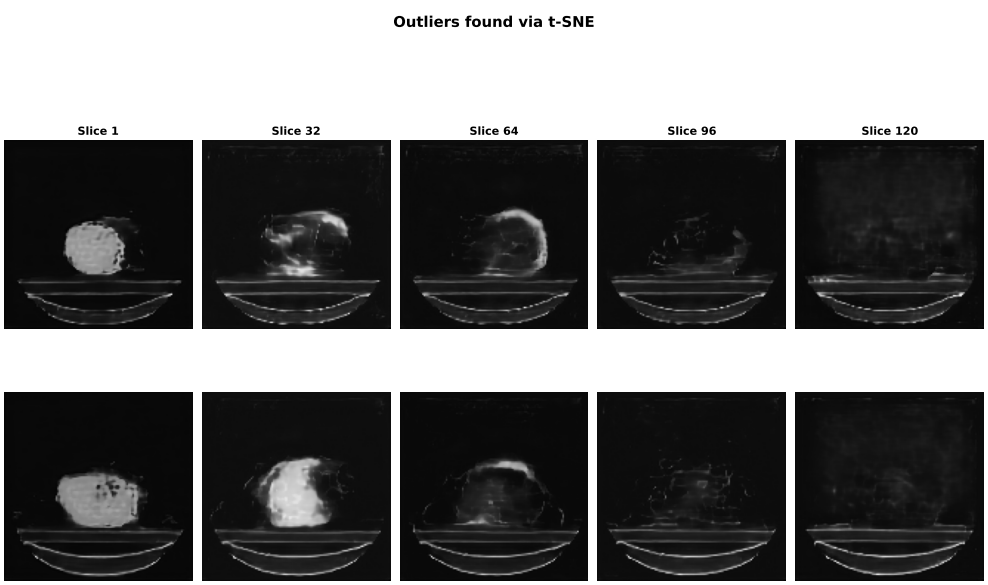

Figure C.3: Examples 12 and 13 of outliers synthetic CT found by t-SNE

Bibliography

[1] Aerts, h. j. w. l., wee, l., rios velazquez, e., leijenaar, r. t. h., parmar, c., grossmann, p., carvalho, s., bussink, j., monshouwer, r., haibe-kains, b., rietveld, d., hoebers, f., rietbergen, m. m., leemans, c. r., dekker, a., quackenbush, j., gillies, r. j., lambin, p. (2014). data from nsclc-radiomics (version 4) [data set]. the cancer imaging archive. https://doi.org/10.7937/k9/tcia.2015.pf0m9rei.

[2] Himanshu Arora. Attention mechanisms in vision models. https://medium.com/jumio/self-attention-in-computer-vision-b929cca5caf8, 2020. 2024-05-27.

[3] David Arthur, Sergei Vassilvitskii, et al. k-means++: The advantages of careful seeding. In *Soda*, volume 7, pages 1027–1035, 2007.

[4] Facundo Bre, Juan Gimenez, and Víctor Fachinotti. Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings*, 158, 11 2017.

[5] T. Brooks, B. Peebles, C. Homes, W. DePue, Y. Guo, L. Jing, D. Schnurr, J. Taylor, T. Luhman, E. Luhman, C. W. Y. Ng, R. Wang, and A. Ramesh. Video generation models as world simulators. `https://openai.com/research/video-generation-models-as-world-simulators`, 2024.

[6] PyTorch Contributors. https://pytorch.org/. 2024-05-27.

[7] Sklearn Contributors. Structural similarity index. https://scikit-image.org/docs/stable/auto_examples/transform/plot_ssim.html. 2024-05-27.

[8] Chris Kuo/Dr. Dataman. Convolutional autoencoders for image noise reduction. https://towardsdatascience.com/convolutional-autoencoders-for-image-noise-reduction-32fce9fc1763, 2019. 2024-05-27.

[9] D. DeMers and G. W. Cottrell. Non-linear dimensionality reduction. In *Advances in neural information processing systems*, volume 5, pages 580–587, 1993.

[10] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

[11] Yuanrui Dong. Resblock — a simple fix helped make deep networks possible. https://medium.com/ai2024-05-27.

[12] Zolnamar Dorjsembe, Sodtavilan Odonchimed, and Furen Xiao. Three-dimensional medical image synthesis with denoising diffusion probabilistic models. In *Medical Imaging with Deep Learning*, 2022.

[13] Zolnamar Dorjsembe, Hsing-Kuo Pao, Sodtavilan Odonchimed, and Furen Xiao. Conditional diffusion models for semantic 3d brain mri synthesis. *IEEE Journal of Biomedical and Health Informatics*, pages 1–10, 2024.

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[15] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks, 2016.

[16] M. Jorge Cardoso et al. Monai: An open-source framework for deep learning in healthcare, 2022.

[17] Lyron Foster. Brain tumor segmentation with u-net in python: A deep learning approach. https://medium.com/@lfoster49203/brain-tumor-segmentation-with-u-net-in-python-a-deep-learning-approach-44758e0812dd, 2023. 2024-05-27.

[18] Richard Garnett. A comprehensive review of dual-energy and multi-spectral computed tomography, 08 2020.

[19] Ibrahim Ethem Hamamci, Sezgin Er, Anjany Sekuboyina, Enis Simsar, Alperen Tezcan, Ayse Gulnihan Simsek, Sevval Nil Esirgun, Furkan Almas, Irem Dogan, Muhammed Furkan Dasdelen, Chinmay Prabhakar, Hadrien Reynaud, Sarthak Pati, Christian Bluethgen, Mehmet Kemal Ozdemir, and Bjoern Menze. Generatect: Text-conditional generation of 3d chest ct volumes, 2024.

[20] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. Brain tumor segmentation with deep neural networks. *Medical Image Analysis*, 35:18–31, 2017.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[23] Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation, 2021.

[24] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2018.

[25] Lan Jiang, Ye Mao, Xi Chen, Xiangfeng Wang, and Chao Li. Cola-diff: Conditional latent diffusion model for multi-modal mri synthesis, 2023.

[26] Vicky Kalogeiton. *Localizing spatially and temporally objects and actions in videos*. PhD thesis, 09 2017.

[27] Firas Khader, Gustav Mueller-Franzes, Soroosh Tayebi Arasteh, Tianyu Han, Christoph Haarburger, Maximilian Schulze-Hagen, Philipp Schad, Sandy Engelhardt, Bettina Baessler, Sebastian Foersch, Johannes Stegmaier, Christiane Kuhl, Sven Nebelung, Jakob Nikolas Kather, and Daniel Truhn. Medical diffusion: Denoising diffusion probabilistic models for 3d medical image generation, 2023.

[28] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[29] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[30] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.

[31] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[32] Jia Leow, Khoh How, Ying Pang, and Hui Yap. Breast cancer classification with histopathological image based on machine learning. *International Journal of Electrical and Computer Engineering (IJECE)*, 13:5885, 10 2023.

[33] Siqi Liu, Sidong Liu, Weidong Cai, Sonia Pujol, Ron Kikinis, and Dagan Feng. Early diagnosis of alzheimer's disease with deep learning. In *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, pages 1015–1018, 2014.

[34] Brian Mwandau and Matunda Nyanchama. *Investigating Keystroke Dynamics as a Two-Factor Biometric Security*. PhD thesis, 06 2018.

[35] Y. V. R. Nagapawan, Kolla Bhanu Prakash, and G. R. Kanagachidambaresan. *Convolutional Neural Network*, pages 45–51. Springer International Publishing, Cham, 2021.

[36] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021.

[37] E. Oja. Data compression, feature extraction, and autoassociation in feedforward neural networks. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, volume 1, pages 737–745. Elsevier Science Publishers B.V., North-Holland, 1991.

[38] OpenAI. Gpt-4 technical report, 2024.

[39] Wei Peng, Ehsan Adeli, Tomas Bosschieter, Sang Hyun Park, Qingyu Zhao, and Kilian M. Pohl. Generating realistic brain mris via a conditional diffusion probabilistic model, 2023.

[40] Walter H. L. Pinaya, Petru-Daniel Tudosiu, Jessica Dafflon, Pedro F da Costa, Virginia Fernandez, Parashkev Nachev, Sebastien Ourselin, and M. Jorge Cardoso. Brain imaging generation with latent diffusion models, 2022.

[41] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[42] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.

[43] Janosh Riebesell. Autoencoder. https://tikz.net/autoencoder/. 2024-05-27.

[44] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[45] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[46] Aditya Sharma. Variational autoencoder in tensorflow. https://learnopencv.com/variational-autoencoder-in-tensorflow/, 2021. 2024-05-27.

[47] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[48] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.

[49] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[50] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[52] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[53] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks, 2018.

[54] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[55] J.J. Weng, N. Ahuja, and T.S. Huang. Learning recognition and segmentation of 3-d objects from 2-d images. In *1993 (4th) International Conference on Computer Vision*, pages 121–128, 1993.

[56] Yuxin Wu and Kaiming He. Group normalization, 2018.

[57] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018.

[58] S. Zhang, J. Liu, B. Hu, et al. Gh-ddm: the generalized hybrid denoising diffusion model for medical image generation. *Multimedia Systems*, 29:1335–1345, 2023.

[59] Junjie Zhao, Donghuan Lu, Kai Ma, Yu Zhang, and Yefeng Zheng. Deep image clustering with category-style representation. 07 2020.

[60] Özgün Cicek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation, 2016.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**