ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF PHYSICS AND ASTRONOMY "A. RIGHI"

SECOND CYCLE DEGREE

PHYSICS

Optimal Transport Driven Flow Matching: An Efficient Framework for Volumetric Generation

Supervisor
Prof. Mirko Degli Esposti

Co-supervisor
Dr. Tommaso Giacometti
Prof. Gastone Castellani

Defended by
Matteo Martelli

Graduation March 2026

Academic Year 2024/2025

Abstract

The development of robust machine learning models for 3D medical imaging is severely hindered by data scarcity, privacy regulations, and dataset heterogeneity. While generative artificial intelligence, particularly Latent Diffusion Models and Flow Matching, represents the state-of-the-art through synthetic data generation, scaling these models to high-dimensional 3D volumes remains a demanding computational challenge. In this thesis, we address the architectural bottlenecks of 3D volumetric generation by optimizing the latent generative core of the MAISI (Medical AI for Synthetic Imaging) framework. We conduct a rigorous theoretical and empirical comparison between Independent Conditional Flow Matching (I-CFM), Variance-Preserving CFM (VP-CFM), and Minibatch Optimal Transport CFM (OT-CFM).

Our extensive evaluation on the CT-RATE dataset reveals a critical "Loss-Fidelity Paradox": while under-parameterized 3D U-Nets achieve similar training convergence across all methods, I-CFM and VP-CFM suffer from severe macroscopic topological failures during inference. We mathematically diagnose this collapse as "Mode Averaging," wherein the network's limited capacity forces it to predict the spatial mean of highly intersecting, conflicting target trajectories, drastically reducing inter-sample diversity. By enforcing Optimal Transport coupling, OT-CFM globally disentangles probability paths, minimizing the variance of the regression target. Quantitative metrics, including Fréchet Inception Distance (FID), Maximum Mean Discrepancy (MMD), and Voxel-wise Variance, confirm that OT-CFM is the only formulation capable of preserving true patient-to-patient anatomical diversity without requiring massive architectural scaling. Ultimately, this work establishes Optimal Transport as an indispensable prerequisite for efficient, high-fidelity 3D medical generation in resource-constrained settings.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Principles of Computed Tomography (CT) Imaging

Computed Tomography (CT) is a medical imaging modality that utilizes X-ray technology to generate detailed cross-sectional images (also called slices) of the body. Since its clinical introduction in the 1970s, CT has evolved into a cornerstone of diagnostic medicine, supplementing conventional radiography by providing three-dimensional volumetric data that eliminates the superposition of anatomical structures seen in 2D X-rays. Unlike a stationary X-ray source used in standard radiography, a CT scanner employs a motorized X-ray tube that rotates rapidly around a circular opening known as the gantry. As the patient is moved through this gantry, the X-ray source emits narrow beams that penetrate the body and are captured by an array of digital detectors positioned directly opposite the source.

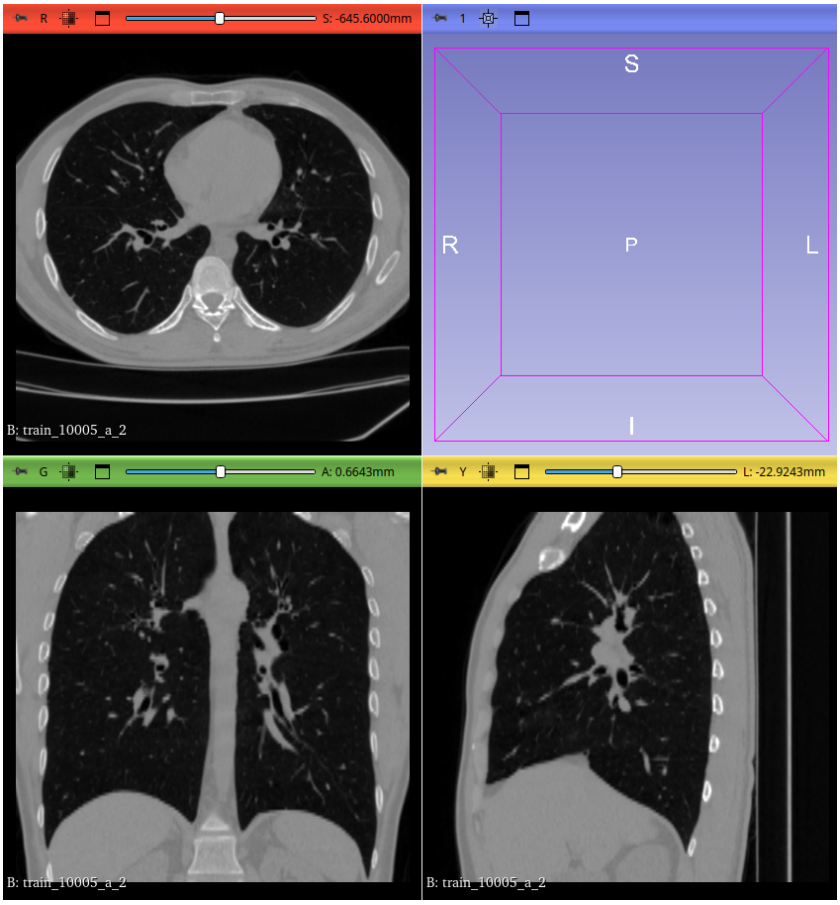

Figure 1: CT scan visualized with 3dSlicer software.

The fundamental principle of CT imaging relies on the different attenuation of X-rays as they pass through biological tissues, a phenomenon governed by the density and atomic number of the material. Dense structures, such as bone, significantly attenuate X-rays and appear bright (hyperdense), while softer tissues like the lungs or fat attenuate less and appear darker (hypodense). These attenuation values are quantified using the Hounsfield Unit (HU) scale [9], a standardized metric where water is defined as 0 HU, air as -1000 HU, and dense bone typically exceeds +1000 HU.

CT is particularly critical in thoracic imaging, serving as the gold standard for evaluating different diseases. It provides an optimal contrast resolution compared to standard radiography that allows a precise detection and characterization of pathologies such as pulmonary nodules, interstitial lung disease, and pneumonia.

1.2 Challenges of Medical Imaging for Machine Learning Models

Medical imaging data presents unique and significant challenges for the development of robust machine learning models, distinguishing it from general purpose computer vision tasks.

First, data scarcity and fragmentation represent a primary challenge. Unlike natural image datasets, large-scale medical datasets are rare due to the high costs and time constraints associated with data acquisition. Furthermore, strict privacy regulations and ethical concerns create "data silos," where hospitals and research centers are often unable to share patient data, preventing the aggregation of large dataset available for the scientific community. Second, medical datasets suffer from intrinsic heterogeneity and domain shift, image quality (e.g resolution) and characteristics can vary drastically across institutions due to differences in hardware (e.g., scanner vendors), acquisition protocols, and reconstruction kernels. This technical variability is compounded by biological variability, as anatomical features and disease manifestations are often heavily dependent on patient characteristics, such as age, gender and other clinical peculiarities. Such inhomogeneity introduces significant biases and makes it difficult for models to generalize across different distributions.

Finally, the reliability of ground-truth labels poses a critical issue. Annotating medical images is a labor intensive process that requires high-level expert knowledge. Consequently, datasets often suffer from label noise due to inter-observer variability discrepancies arising from the subjective interpretation of different radiologists and intra observer inconsistency. Additionally, medical datasets are frequently plagued by class imbalance, where pathological cases are unbalanced compared to healthy controls, further complicating the training of discriminative models.

In this context, generative artificial intelligence is a useful solution to overcome the limitations of data scarcity and heterogeneity. Generative models, during the training, focus on learning the underlying data distribution of the data itself by approximating the high-dimensional space on which real medical images lie, these models can sample new, synthetic instances that preserve the statistical properties and anatomical realism of the original data, without corresponding to any specific real-world patient. This powerful method allows to face the problems discussed before:

- Synthetic data is generated by sampling from a learned distribution rather than retrieving existing records, it breaks the one to one mapping with real patients. This allows for the sharing of realistic, high fidelity datasets across institutions without violating privacy regulations.

- Advanced generative frameworks can be employed to map images from one domain (e.g., a specific scanner protocol) to another. This helps in standardizing heterogeneous datasets, reducing the domain shift caused by technical variability in acquisition hardware.

- Generative models can produce an unlimited number of synthetic samples, effectively enriching training datasets. This is particularly crucial for addressing class imbalance.

1.3 Generative Modeling for Medical Imaging

While early approaches relied heavily on Generative Adversarial Networks (GAN), these architectures often suffer from training instability and mode collapse, failing to capture the full diversity of the data distribution. Consequently, the field has shifted towards likelihood-based models (CNFs), diffusion models and, more recently, Flow Matching methods.

Standard Diffusion Models are governed by the stochastic differential equation (SDEs), these methods produce curved probability paths that lead complex vector fields requiring a large number of integration steps and resulting in slow inference times. Flow matching methods propose a generalized framework based on Ordinary Differential Equation (ODE) which has the ability to incorporate the Diffusion models formulations and provide the most efficient way to define a continuous time dynamic that transports a simple base distribution (e.g., Gaussian noise) to the complex empirical data distribution along predefined probability paths.

Flow matching constructs vector fields that are significantly easier for neural networks to approximate. In fact, this approach shows higher sample fidelity and faster training and sampling, making them the state of the art choice for image generation in recent years 2.

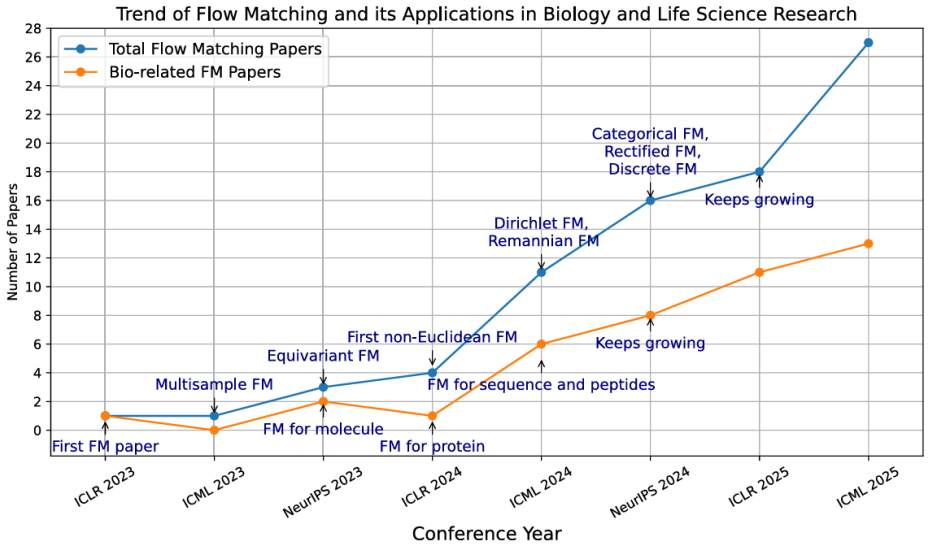

Figure 2: Trend of published papers on flow matching and its applications in biology and life sciences across major machine learning conferences from 2023 to 2025 [1].

1.3.1 Flow optimization: Minibatch Optimal Transport

In standard Flow Matching, samples drawn from the base distribution (Gaussian noise) and the target distribution (real data) are coupled purely at random. In high-dimensional spaces, such as those required for 3D medical volume generation, this independent coupling generates probability paths that can intersect frequently. These intersections produce a vector field with high spatial variance, which is complex for a neural network to approximate during the sampling phase.

To address this structural limitation, the theory of Optimal Transport (OT) is introduced and dynamically applied via Minibatch Optimal Transport. The Optimal Transport problem consists of finding an assignment plan between two distributions that minimizes the total "cost" of moving mass, typically measured through the squared Euclidean distance (closely related to the Wasserstein-2 distance).

By enforcing minimum-cost couplings, OT generates trajectories that are globally more coherent and reduce unnecessary intersections. The resulting vector field becomes exceptionally smooth and regular. This regularity guarantees faster convergence during training and allows the Ordinary Differential Equation (ODE) to be solved during inference using numerical solvers with lower number of steps. This effectively reduces the computational overhead and improves the stability of the generation, making high-fidelity volumetric generation scalable.

1.4 Latent Diffusion and MAISI Framework

To address the computational constraints imposed by high dimensional 3D data such as CT scans, this thesis adopts the paradigm Latent Flow Matching, inspired by Latent Diffusion Models (LDMs) [10].

Unlike standard generative models that operate directly in the high-dimensional pixel space, with this approach, before proceeding with the generative phase, we have a compression stage, usually a Variational AutoEncoder (VAE), that maps the input images in a lower dimensional latent space. Then, the generative process is performed within this compressed representation.

This compression from pixel space to latent space drastically reduces the computational resources required for training and inference, and it allows the model to focus on the semantic structure of the data rather than imperceptible high-frequency details.

In this work, we specifically leverage the MAISI (Medical AI for Synthetic Imaging) framework [6]. Developed by the MONAI consortium and NVIDIA, MAISI represents a state-of-the-art open-source initiative designed to establish a foundation model for medical imaging. It provides a modular and scalable architecture capable of handling the high dimensionality and multi-modal nature of medical data (e.g., CT, MRI) through advanced latent generation techniques. The primary objective of this thesis is to analyze the MAISI architecture and implement a computationally efficient optimization of its generative core. Our goal is to develop a "lighter" version of the generative model that reduces the training computational overhead while preserving, as much as possible, the high fidelity and anatomical correctness of the generated 3D volumes. By lowering the hardware barrier to entry, this approach aims to democratize access to advanced generative tools, enabling smaller research groups with limited computational resources to leverage 3D synthetic data generation.

1.5 Objectives and Contributions

The primary objective of this thesis is to theoretically analyze the mathematical generative framework and implement a computationally optimized generation of 3D volumes. Specifically, the core contributions are structured as follows:

- **Theoretical Foundation (Chapter 2):** We provide a comprehensive theoretical analysis of Flow Matching. We examine its mathematical formulations, detailing the differences between independent coupling and Optimal Transport driven approaches, and compare them with standard diffusion paradigms.
- **Framework Analysis (Chapter 3):** We critically evaluate the MAISI framework, tracing its architectural evolution from Denoising Diffusion Probabilistic Models (DDPMs) to the recent adoption of a flow matching method called 1-Rectified Flow. We investigate its latent generative core to identify some potential areas for structural improvement.
- **Optimized Implementation (Chapter 4):** We implement a completely new model for the generative dynamics within the MAISI framework with a highly optimized Minibatch Optimal Transport Conditional Flow Matching (OT-CFM) architecture. We present the core part of the code discussing the implementation details, our architectural choices, and the software engineering required to integrate this ODE-based framework into the existing pipeline.
- **Evaluation and Results (Chapter 5):** Finally, we conduct a quantitative and qualitative evaluation of the generated 3D synthetic volumes. We compare the performance of our optimized model against the baselines, discussing the concrete improvements in computational efficiency while preserving anatomical fidelity.

2 Latent Diffusion models

2.1 Variational AutoEncoder (VAE)

As motivated in the first chapter, The first stage of a Latent Diffusion Model is the compression of data from the high-dimensional pixel space into a compact, semantically rich latent space. To achieve this, we implement a Variational Autoencoder (VAE) [11].

In machine learning, a Variational AutoEncoder is a type of neural network architecture, which is composed of a combination of a classical Autoencoder (fig. 3) and variational inference techniques. An autoencoder learns an encoding function that maps input data to a lower-dimensional latent representation, and a decoding function that reconstructs the input from this representation, typically for dimensionality reduction or feature learning.

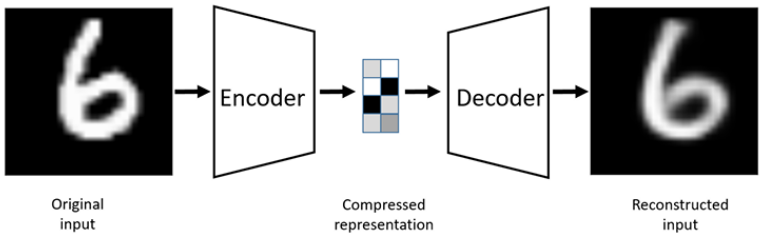

Figure 3: An autoencoder example. The input image is encoded to a compressed representation and then reconstructed through a decoder [2].

Unlike classical Autoencoders, which learn a deterministic mapping $x \to z \to x'$, a Variational Autoencoder [11] is a generative model based in Bayesian inference. Ideally, given an input image x, we can compute the true posterior distribution $p(z|x)$ of the latent variables. According to Bayes' theorem:

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x|z)p(z)}{\int p(x|z)p(z)dz} \tag{1}$$

However, computing the marginal likelihood $p(x)$ requires integrating over the entire high-dimensional latent space, which is analytically intractable for complex data distributions. The VAE addresses this by introducing a parametric approximation of the posterior distribution, instead of computing the true posterior $p(z|x)$ exactly, we introduce a parametric approximation $q_\phi(z|x)$, typically chosen to be a multivariate Gaussian distribution. Formally, the VAE consists of two coupled probabilistic networks:

- An Encoder (or inference model) $q_\phi(z|x)$, which approximates the true posterior $p(z|x)$ by mapping the input data x to a distribution over the latent space.

- A Decoder (or generative model) $p_\theta(x|z)$, which reconstructs the data distribution given a latent sample z.

To train the model, we aim to maximize the log-likelihood of the data $\log p_\theta(x)$. Since this is intractable, we maximize a lower bound known as the Evidence Lower Bound (ELBO) [12]. The objective function can be derived as:

$$\log p_\theta(x) \geq \mathcal{L}_{\text{ELBO}}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z)) \tag{2}$$

This loss function is composed of two competing terms with distinct physical interpretations:

- The Reconstruction Loss $\mathbb{E}_{q_\phi}[\log p_\theta(x|z)]$: This term encourages the decoder to reconstruct the input data from the latent representation z, minimizing the distortion (typically implemented as MSE or L1 loss).
- The Regularization Term $D_{KL}(q_\phi(z|x)||p(z))$: This term measures the divergence between the learned posterior and the fixed prior $p(z) = \mathcal{N}(0, I)$. In the case of Gaussian distributions, this loss function can be conceptually decomposed into two distinct regularization forces acting on the latent parameters:
  - Centering Force (mean μ): The first component of the loss minimizes the squared norm of the mean vectors (μ^2). Geometrically, this acts like a spring pulling the latent representations of all data points towards the origin of the coordinate system. This prevents the encoder from scattering clusters far apart in the latent space, ensuring compactness.
  - Variance Constraint (variance σ^2): The second component regulates the spread (uncertainty) of the distribution. It penalizes variances that differ from unity. Without this constraint ($\sigma^2 \to 0$), the encoder could learn to map each input image to a specific, infinitesimal point in space (a Dirac delta), while this minimizes reconstruction error, it results in pure memorization, degenerating in a pure autoencoder. The latent space would be full of "holes" or undefined regions between points. Sampling from these empty regions would produce garbage output, making the model useless for generation.

    With this constraint ($\sigma^2 \approx 1$), the model is forced to map each input not to a point, but to a probability volume (a "cloud") in the latent space. This ensures that the latent representations of similar images overlap. This overlap forces the decoder to ensure that points between two examples decode to a coherent mix of the two, ensuring the continuity of the latent space[13].

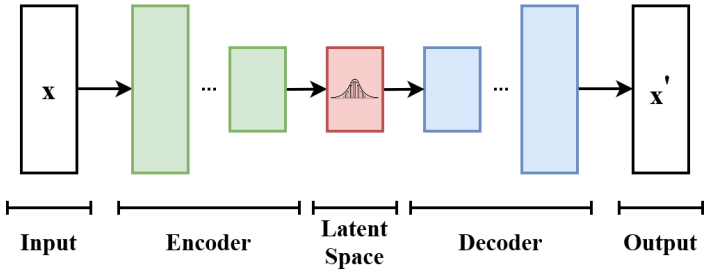

Figure 4: Scheme of a Variational Autoencoder. We have a gaussian shape of the latent space due to the regularization term (KL divergence).

2.1.1 The Reparameterization Trick

A critical challenge arises when training the VAE architecture described above. To update the encoder parameters ϕ via gradient descent (backpropagation), we need to compute the gradient of the loss function $\nabla_\phi \mathcal{L}_{\text{ELBO}}$.

However, the expectation in Equation 2 involves sampling the latent vector z from the posterior distribution $z \sim \mathcal{N}(\mu_\phi(x), \sigma^2_\phi(x))$.

Sampling is inherently a stochastic operation and is non-differentiable; gradients cannot flow through a random node in the computational graph. This would disconnect the encoder from the decoder, making end-to-end training impossible. To address this, Kingma and Welling introduced the reparameterization trick (fig. 5).

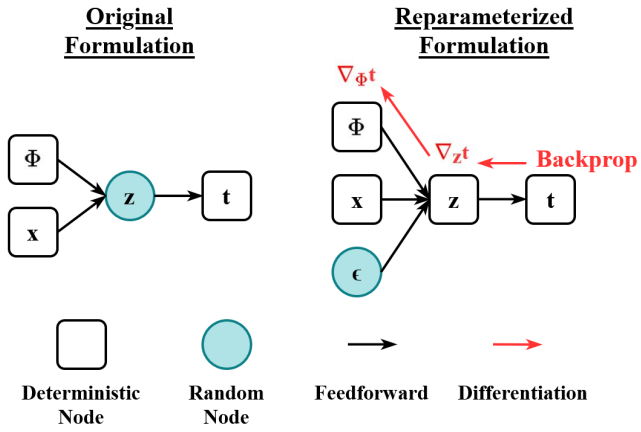

Figure 5: Scheme of the reparameterization trick. The random variable ϵ is injected into the latent space z as external input. In this way, it is possible to backpropagate the gradient without involving stochastic variable during the update.

The key idea is to externalize the randomness by expressing the random variable z as a deterministic transformation of the distribution parameters (μ, σ) and an auxiliary independent noise variable ϵ. Instead of sampling z directly, we sample ϵ from a standard normal distribution:

$$\epsilon \sim \mathcal{N}(0, I_d) \tag{3}$$

and then construct z via the transformation:

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon \tag{4}$$

where $\odot$ denotes the element-wise product.

This reformulation transforms the sampling process. In the computational graph, z is now a deterministic function of ϕ (via μ and σ) and a fixed input ϵ (fig. 6). The stochasticity is injected as an input, not as an operation. Crucially, this allows gradients to flow backwards through the deterministic nodes μ and σ to the encoder weights, enabling the optimization of the entire network using standard stochastic gradient descent.

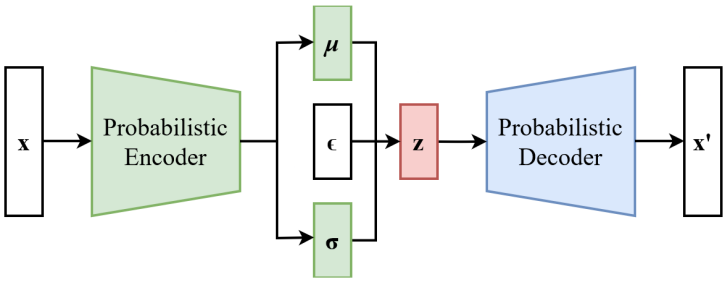

Figure 6: Scheme of a variational autoencoder after the reparameterization trick.

2.2 Continuous Time Generative Dynamics

Having established the VAE as a method to compress high-dimensional data into a continuous latent space, the next challenge is to model the distribution of these latent codes to generate new samples. While the VAE prior is typically Gaussian, the aggregate posterior of the encoded data is often complex and multimodal. To bridge the gap between a simple noise distribution and the complex data distribution, we turn to continuous-time generative models. These models define the generative process not as a single step, but as a dynamic evolution governed by differential equations. In the following sections, we introduce Neural Ordinary Differential Equations (NODEs) to describe the dynamics and Continuous Normalizing Flows (CNFs) to track the evolution of probability densities.

2.2.1 Neural Ordinary Differential Equation (NODE)

Consider the following ordinary differential equation (ODE):

$$\frac{dy}{dt}(t) = f(t, y(t)) \tag{5}$$

with initial condition $y(0) = y_0$. If the function f is described by a neural network, the differential equation can be rewritten to express the dependency on the parameters θ of the network as:

$$y(0) = y_0, \frac{dy}{dt}(t) = f_\theta(t, y(t)) \tag{6}$$

with $y_0 \in \mathbb{R}^{d_1 \times \ldots \times d_k}$ a tensor (any-dimensional) describing the initial condition of the system, $f_\theta(t, y(t)) : \mathbb{R} \times \mathbb{R}^{d_1 \times \ldots \times d_k} \to \mathbb{R}^{d_1 \times \ldots \times d_k}$ function specified by a neural network and the list of parameters of the network f_θ This is called Neural Ordinary Differential Equation (NODE). The solution of the NODE is a function $y(t) : [0, 1] \to \mathbb{R}^{d_1 \times \ldots \times d_k}$ which usually cannot be obtained analytically (at least when the neural network is not particularly simple): the workaround is to use numerical differential equation solvers, which have become increasingly fast and reliable to obtain a numerical approximation of the function itself. The most simple solver method for Equation 6 is the Euler method: the solution can be computed by replacing the derivative with the finite difference:

$$\frac{y(t+h) - y(t)}{h} = f_\theta(t, y(t))$$

and so, with fixed step size h, it is possible to iteratively compute $y(t)$ starting from $y(0)$ by using

$$y_{n+1} = y_n + h f_\theta(t_n, y_n)$$

with $t_n = t_0 + nh$ and $y_n = y(t_n)$ being an estimate of the exact solution at time t_n. Numerical solvers for differential equations still incur approximation errors of the real

solution: when using the Euler method, the error w.r.t. the real solution is governed by the step size h, with smaller steps leading to a more accurate solution. On the other hand, small steps quickly become a computational burden when the solution is to be computed, requiring many more iterations to reach the time of the solution.

To intuitively understand the conceptual shift introduced by NODEs, it is useful to compare them with traditional Residual Networks (ResNets). A standard ResNet updates the hidden state through discrete transformations, formulated as $y_{n+1} = y_n + f_\theta(y_n)$. Mathematically, this is strictly equivalent to a single step of the Euler discretization method with a fixed step size $h = 1$. As illustrated in Figure 7, while a ResNet applies a discrete sequence of finite transformations (left), an ODE network generalizes this paradigm to the continuous limit (right). Instead of learning discrete layer weights, the neural network parameterizes a continuous vector field, allowing the state to evolve smoothly over time. This continuous formulation not only provides a more elegant mathematical framework but also delegates the choice of evaluation locations (the black circles in the figure) to the numerical solver, which can dynamically adjust them based on the desired error tolerance.

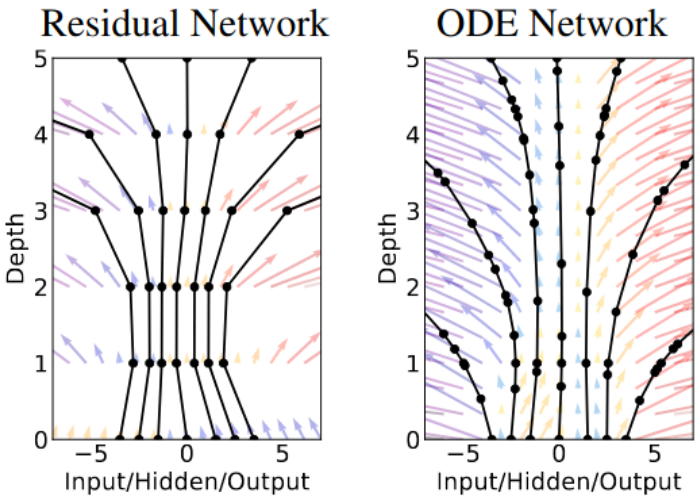

Figure 7: Left: A Residual network defines a discrete sequence of finite transformations. **Right:** An ODE network defines a vector field, which continuously transforms the state. Both: Circles represent evaluation locations (image from [3]).

2.2.2 Continuous Normalizing Flows (CNF)

Continuous Normalizing Flows (CNFs) are used to transform, through a series of continuous transformations, an input distribution (e.g., Gaussian) into an arbitrary target distribution. As visually represented in Figure 8, this generative process allows a simple, tractable base probability density (such as a standard Gaussian at t_0) to smoothly evolve over time into a highly non-linear, multimodal data distribution at t_1. This topological transformation is entirely guided by the continuous vector field parameterized by the neural network.

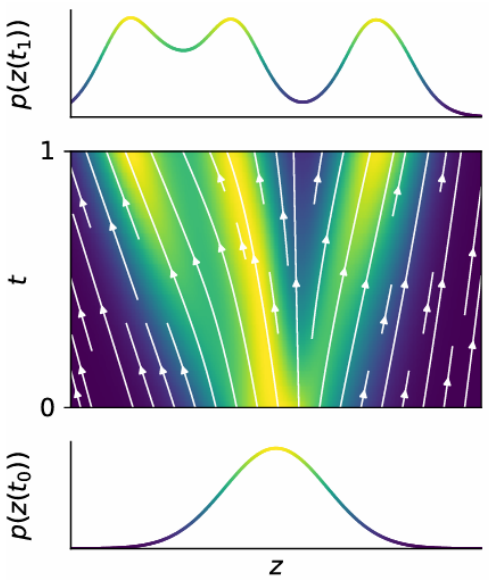

Figure 8: The Gaussian distribution at t_0 is transformed in a more complex distribution with continuous dynamics learned by the neural network. Image from [4]

The main advantage of continuous normalizing flows w.r.t. discrete normalizing flows is that the invertibility of the transformation is guaranteed by the uniqueness of the ODE solution. This implies that the defining dynamics f_θ does not need to be invertible itself, nor constrained to have a triangular Jacobian, allowing for arbitrary neural network architectures to parameterize the vector field.

Suppose we observe a distribution $\mathbb{P}$ and wish to learn its approximation having available only some samples and also consider the Neural Ordinary Differential Equation [3] (NODE):

$$y \sim \mathcal{N}(0, I_{d\times d}), \quad \frac{dy}{dt}(t) = f_\theta(t, y(t)) \quad \text{for } t \in [0, T] \tag{7}$$

The goal is to train the model in order to have a final distribution $y(T)$ that approach the target $\mathbb{P}$ with a good approximation. The differential equation describes a flow

from the initial probability distribution $y(0) = \mathcal{N}(0, I_{dxd})$ to the final $y(T)$. The flow is trained by maximizing the log-likelihood, in order to do so we define of y as a function of time and $p_\theta(t, y(t))$ be the density of $y(t)$ for each time $t \in [0, T]$. The change in log probability density is governed by the differential equation:

$$\frac{\partial \log p_\theta(t, y(t))}{\partial t} = -\sum_{k=1}^{d} \frac{\partial f_{\theta,k}(t, y(t))}{\partial y_k} = -Tr\left(\frac{df_\theta(t, y(t))}{dy(t)}\right) \tag{8}$$

While the trace of the Jacobian allows for likelihood maximization, its evaluation cost is $\mathcal{O}(d^2)$, which is prohibitive for high dimensional data. To address this, the Hutchinson's trace estimator is introduced [4] to reduce the cost to $\mathcal{O}(d)$:

$$Tr(A) = \mathbb{E}_\epsilon[\epsilon^T A \epsilon] \tag{9}$$

where ϵ is a random variable with $\mathbb{E}[\epsilon] = 0$ and $Cov[\epsilon] = I$.

Maximizing the likelihood for N empirical samples corresponds to minimizing the negative log-likelihood. Integrating over time $[0, T]$, we obtain the loss function:

$$-\frac{1}{N}\sum_{i=1}^{N} \log p_\theta(T, y_i) = -\frac{1}{N}\sum_{i=1}^{N}\left[\log p_\theta(0, y(0, y_i)) - \mathbb{E}_\epsilon\left[\int_0^T \epsilon^T \frac{\partial f_\theta}{\partial y}\epsilon \, dt\right]\right] \tag{10}$$

The final training objective can be expressed as:

$$\mathcal{L}(\theta) = -\mathbb{E}_{y\sim\pi}[\log p_\theta(T, y)] \tag{11}$$

which is the expectation value of the learned parametric distribution at time T over the samples taken from the target distribution $y(T)$.

In order to evaluate the loss, equation 10 needs to be computed and $y(t, y_i)$ is to be known from 0 to T in order to be integrated. In order to do this, given a sample $y_i \in \mathbb{R}$, equation 7 is evaluated backwards in time from $t = 1$ to $t = 0$ allowing for the evaluation of the integral. Afterward equation 11 can be backpropagated in order to update the weights of the model, progressing in time from $t = 0$ to $t = 1$.

The main problem of this approach is that in each step in training, the loss function is evaluated to update the weights, which in turn requires solving an ODE simulation with numerical approximation methods. This is expensive both in terms of training time and computational resources. Furthermore, as training progresses, the learned dynamics often become stiff, requiring numerical solvers to take increasingly smaller steps, further exacerbating the computational burden. So this method becomes impracticable when the dimensionality of the problem grows, no scalable CNF training algorithms are known [14].

For this reason, we now introduce different techniques which do not require ODE integration at each training step (i.e. "simulation free") called Flow Matching.

2.3 Flow Matching

Our first goal is to approximate an unknown data distribution, denoted as $q(x_1)$, we assume that we can only have access to some data samples but the density function itself remain unknown. The Flow Matching framework addresses this by defining a time dependent probability density path p_t over the interval $t \in [0, 1]$. This path is constructed to model our starting distribution, usually the standard normal distribution $p(x) = \mathcal{N}(x|0, I)$, into a final distribution p_1 that approximates the target data distribution $q(x_1)$.

The evolution of this probability density is governed by a time dependent vector field $u_t(x)$. The core principle of Flow Matching is to train a neural network, parameterized by θ and denoted as $v_t(x; \theta)$, to regress this target vector field. The training objective is defined as minimizing the expected mean squared error between the learned vector field and the target field over the path:

$$\mathcal{L}_{FM}(\theta) = \mathbb{E}_{t,p_t(x)}||v_t(x) - u_t(x)||^2 \tag{12}$$

where θ denotes the learnable parameters of the CNF vector field v_t, $t \sim \mathcal{U}[0, 1]$ and $x \sim p_t(x)$.

By minimizing $\mathcal{L}_{FM}$ loss, the neural network learns the flow dynamics. Consequently, at inference time, we can generate samples of the target distribution p_1 starting from the prior distribution p_0 passing through the numerical integration of the learned vector field v_t [14].

The problem with this initial formulation of the Flow Matching is that is not possible to use it in practice since we have no prior knowledge for what an appropriate p_t and u_t are. There are many choices of probability paths that can satisfy $p_1(x) \approx q(x)$, and more importantly, we generally do not have access to a closed form u_t that generates the desired p_t. This lack of closed form targets prevents the direct computation of gradients for optimization.

2.4 Conditional Flow Matching (CFM)

To overcome the Flow matching limitation described in the previous section, the problem must be reformulated using conditional probability paths and conditional vector fields, which allow for a tractable, per sample definition of the flow.

Instead of modeling the intractable marginal probability path $p_t(x)$ directly, the Conditional Flow Matching framework proposes to construct it as a mixture of simpler conditional paths.

To achieve this, we have to introduce a latent conditioning variable z distributed according to $q(z)$. The choice of z and its distribution $q(z)$ is one of the characteristics that distinguishes the different Flow Matching methods. In the standard formulation described by Lipman et al. [14], this variable represents a data sample ($z = x_1$), whereas in generalized frameworks like Tong et al. [15], it can represent a pair of source and target samples ($z = (x_0, x_1)$). We will return into these distinctions in greater depth in the following sections.

We define a conditional probability path $p_t(x|z)$ starting from $p_0(x|z)$ and ending at $p_1(x|z)$. The marginal probability path $p_t(x)$ is then recovered by marginalizing over the latent variable z:

$$p_t(x) = \int p_t(x|z)q(z)dz \tag{13}$$

In other words, we can construct the marginal probability path as an aggregation of simple conditional probability paths.

Similarly, we define a conditional vector field $u_t(x|z)$ that generates the specific conditional path $p_t(x|z)$. The key insight of the CFM framework is that the global (marginal) vector field $u_t(x)$, which we originally sought to regress in the FM objective, can be expressed as an aggregation of these conditional vector fields (fig. 9) (dim. in appendix A):

$$u_t(x) = \mathbb{E}_{q(z)}\left[\frac{u_t(x|z)p_t(x|z)}{p_t(x)}\right] \tag{14}$$

While computing this marginal vector field remains intractable due to the integral involved, Lipman et al. [14] and Tong et al. [15] demonstrated that it is possible to bypass its explicit computation. They introduced the Conditional Flow Matching (CFM) objective, which regresses the neural network $v_t(x;\theta)$ directly against the conditional vector field $u_t(x|z)$:

$$\mathcal{L}_{CFM}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],z\sim q(z),x\sim p_t(x|z)}\left[||v_t(x;\theta) - u_t(x|z)||^2\right] \tag{15}$$

A fundamental theorem proves (dim in appendix B) that the gradients of the CFM objective with respect to the parameters θ are equivalent to the gradients of the intractable FM objective:

$$\nabla_\theta\mathcal{L}_{CFM}(\theta) \approx \nabla_\theta\mathcal{L}_{FM}(\theta) \tag{16}$$

This equivalence implies that minimizing $\mathcal{L}_{CFM}$ effectively minimizes $\mathcal{L}_{FM}$.

Unlike the marginal vector field, the conditional vector field $u_t(x|z)$ can be chosen to have a simple, closed-form solution (e.g., a linear trajectory), making the training objective tractable and efficient. The neural network, by seeing many simple conditional flows during training, implicitly learns to approximate the complex marginal vector field required for generation.

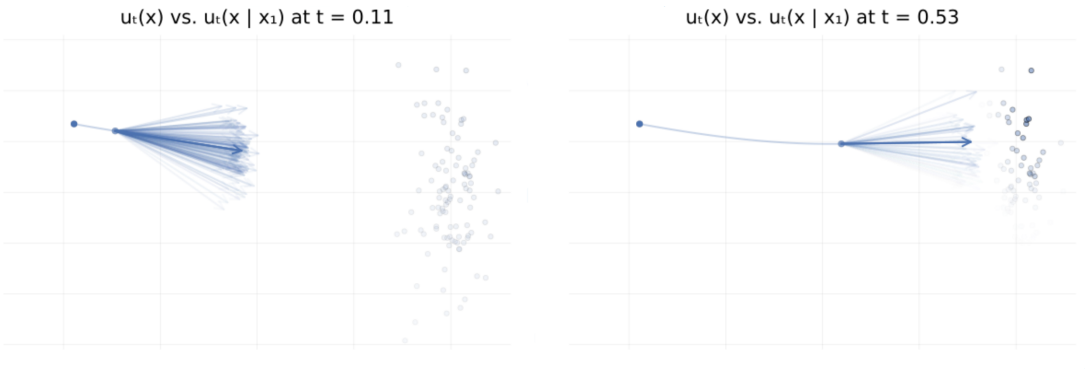

Figure 9: Marginal vector field $u_t(x)$ (which is computed as the expectation (average) of the conditional fields) vs. conditional vector field $u_t(x|x_1)$ for samples $x_1 \sim p_1$. Here $p_0 = p_1 = \mathcal{N}(0, 1)$ and the two trajectories are according to the marginal vector field $u_t(x)$. The neural network's objective is to approximate this global average by learning from the individual simple conditional directions. Image from [5].

To facilitate the understanding of the mathematical framework used in the following sections, we summarize the key notations and their physical interpretations in Table 1.

Symbol	Mathematical Definition	Physical Interpretation
$q(x_1)$	Target Data Distribution	The empirical distribution of the real dataset (fixed).
$q(x_0)$ / $p_0(x)$	Source Distribution	The starting distribution (e.g., Gaussian noise or a source dataset).
$p_t(x)$	Marginal Probability Path	The time-dependent density evolving from source to target (dynamic).
x_0, x_1	Samples from $q(x_0), q(x_1)$	Individual points: initial source and final target.
z	Latent conditioning variable	The specific instruction or "code" defining the current trajectory.
$q(z)$	Mixing distribution	The rule for pairing source and target samples (e.g., independent vs. OT).
$p_t(x\|z)$	Conditional probability path	A simple, tractable distribution (e.g., Gaussian) moving from x_0 to x_1.
$u_t(x\|z)$	Conditional vector field	The velocity field generating the specific path for z (e.g., constant velocity).
$u_t(x)$	Marginal vector field	The global average velocity field (intractable target for regression).
$v_t(x;\theta)$	Neural Network	The parameterized model trained to approximate $u_t(x\|z)$.

Table 1: Summary of the mathematical notation used in the Conditional Flow Matching framework. Note the convention where $q(\cdot)$ generally refers to fixed boundary distributions (source/target), while $p_t(\cdot)$ refers to the evolving path modeled by the flow.

2.5 Theoretical Foundations

This section establishes the mathematical groundwork of Conditional Flow Matching (CFM). We first present the original formulation based on Gaussian probability paths (Lipman et al.), analyze the geometric properties of optimal flows via kinetic energy minimization, and demonstrate how Diffusion Models arise as a specific instance of this framework.

2.5.1 Gaussian Probability Paths

Crucially, the vector field $u_t(x|z)$ regressed by the neural network is not arbitrary but is the direct consequence of the chosen probability path $p_t(x|z)$. Following the derivation in Lipman et al., we consider a Gaussian conditional path defined by time-dependent mean $\mu_t(z)$ and standard deviation $\sigma_t(z)$:

$$p_t(x|z) = \mathcal{N}(x; \mu_t(z), \sigma_t(z)^2 I) \tag{17}$$

To find the unique vector field generating this path, we utilize the *reparameterization trick*. A sample $x(t)$ from this path can be expressed as a deterministic transformation of a noise variable $\epsilon \sim \mathcal{N}(0, I)$:

$$x(t) = \mu_t(z) + \sigma_t(z) \cdot \epsilon \tag{18}$$

The velocity field $u_t(x|z) = \frac{dx}{dt}$ describing the flow is obtained by differentiating the sample position with respect to time:

$$\frac{dx}{dt} = \dot{\mu}_t(z) + \dot{\sigma}_t(z) \cdot \epsilon \tag{19}$$

By inverting the position equation to express the noise as $\epsilon = \frac{x-\mu_t(z)}{\sigma_t(z)}$ and substituting it back, we obtain the closed-form analytical vector field:

$$u_t(x|z) = \dot{\mu}_t(z) + \frac{\dot{\sigma}_t(z)}{\sigma_t(z)}(x - \mu_t(z)) \tag{20}$$

This equation shows that for Gaussian paths, the vector field is a linear combination of a transport term ($\dot{\mu}_t$) and an expansion/contraction term dependent on the noise schedule ($\dot{\sigma}_t/\sigma_t$).

2.5.2 Geometric Optimality: Minimizing Kinetic Energy

We have seen that the gaussian formulation provides a closed-form solution for the vector field, but now we must ask: which is the most efficient path to transport the distribution from the initial to the final state?

The efficiency of a generative flow is formalized by the Benamou-Brenier energy functional, which measures the total effort of the transport plan:

$$\mathcal{E}_{kin}(v) = \int_0^1 \int_{\mathbb{R}^d} \frac{1}{2}||v_t(x)||^2 p_t(x) dx dt \tag{21}$$

Trajectories that minimize this kinetic energy correspond to geodesics in the Wasserstein space. In Euclidean space, the optimal transport plan between two points is a straight line traversed at constant velocity [16][17][18].

$$\psi_t(x_0, x_1) = (1-t)x_0 + tx_1 \tag{22}$$

Therefore, a flow is considered optimal if it exhibits zero acceleration ($\ddot{x} = 0$) and straight trajectories. As we we are going to discuss, even if the mean μ_t follows a straight line, the expansion term $\dot{\sigma}/\sigma$ in equation 20 introduces non-constant velocity components (expansion and contraction) that increase the total kinetic energy and curvature of the flow.

2.5.3 Lipman's Formulation: The Stochastic Compromise

Lipman et al. utilized the Gaussian framework (sec. 2.5.1) to approximate the objective (sec. 2.5.2), the primary goal of his work is to create a probabilistic framework to unify Flow Matching and Diffusion Models.

Since Diffusion Models are generative models that learn to map pure noise to data without paired supervision, they oriented the problem as generating x_1 (data) without a known source. So Lipman et al. constrained the conditioning variable z to only the target sample

$$z = x_1 \tag{23}$$

This design choice has a major geometric consequence. Since the source x_0 is marginalized (unknown), the conditional path $p_t(x|x_1)$ must be a a probability density "cloud" capable of covering all possible noise sources that could generate x_1 (fig. 10). This choice lead a compromise:

- the Mean (μ_t): Defined to approximate Optimal Transport, following the linear interpolant towards the target: $\mu_t(x_1) = tx_1$ (assuming centered noise).
- The Variance (σ_t): Since the source x_0 is unknown, the path cannot be deterministic ($\sigma = 0$). It must be a probability cloud covering all possible sources. Thus, a non-zero noise schedule $\sigma_t(x_1) > 0$ is compulsory.

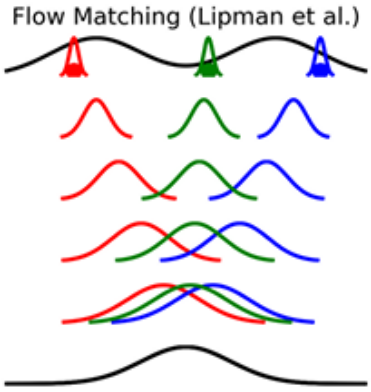

Figure 10: Gaussian path Conditional Flow Matching.

So we have a probability path defined as:

$$p_t(x|x_1) = \mathcal{N}(x; tx_1, \sigma_t^2 I) \tag{24}$$

and substituting these choices into Eq. 20, we get Lipman's specific vector field:

$$u_t(x|x_1) = \underbrace{x_1}_{\text{Target Velocity}} + \underbrace{\frac{\dot{\sigma}_t}{\sigma_t}(x - tx_1)}_{\text{Noise Correction}} \tag{25}$$

There is the possibility to further improve the performance of this framework by removing the noise correction term, as we will discuss in Section 2.6. However, it is first important to complete the description by highlighting the link between Lipman's framework and Diffusion Models.

2.5.4 Diffusion Models as a Gaussian Subset

This framework allows us to formally categorize Diffusion Models (DMs) not as a separate entity, but as a specific, constrained instance of Flow Matching framework. Score-based generative models are defined by a Variance Preserving (VP) SDE. If we map the DM noise schedule to the CFM parameters:

$$\mu_t(x_1) = \alpha_t x_1, \quad \sigma_t(x_1) = \sqrt{1 - \alpha_t^2} \tag{26}$$

and substitute these into Eq. 20, the CFM vector field exactly recovers the Probability Flow ODE of diffusion (derivation in appendix C):

$$u_t(x|x_1) = -\frac{1}{2}\beta_t[x + \nabla \log p_t(x|x_1)] \tag{27}$$

This proves that Diffusion Models are mathematically a subset of Conditional Flow Matching where the probability path is constrained to be Gaussian and the parameters μ_t, σ_t follow a diffusion schedule. While theoretically sound, these paths are geometrically sub-optimal because they follow curved trajectories with non-constant velocity, leading to higher kinetic energy and slower sampling (requiring more NFE) (fig. 11).

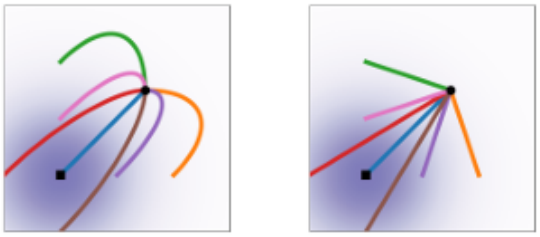

Figure 11: Left: Curved paths obtained with diffusion schedules. Right: Straight paths obtained with optimal transport.

2.6 Independent Conditional Flow Matching (I-CFM)

The limitation of the gaussian formulation (and Diffusion Models) lies in the intrinsic uncertainty of equation 20. If we attempt to enforce deterministic paths by setting $\sigma \to 0$, the term $\frac{\dot{\sigma}}{\sigma}$ explodes unless the sample x is perfectly aligned with μ_t.

To overcome this limitation Tong et al. observed that in the CFM objective (eq. 15) we have no particular constraints regarding the choice of z. We are free to condition the latent variable z not only on the target distribution $z = x_1$ but also on a specific sample x_0 of the starting distribution, so the conditioning variable becomes $z = (x_0, x_1)$. This approach provides a way to model the flow between any arbitrary source distribution $q(x_0)$ and the target data distribution $q(x_1)$ without requiring analytical knowledge of the source density (fig. 12).

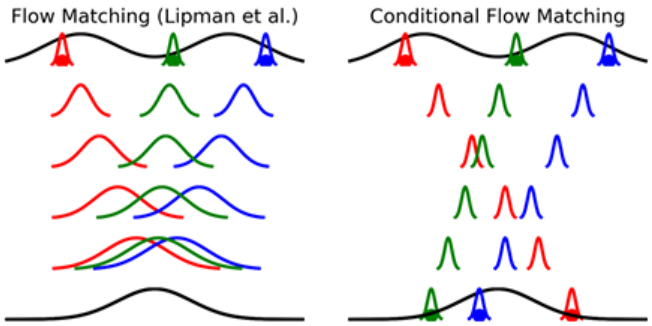

Figure 12: Left: Gaussian paths Conditional Flow Matching, Right: Independent Conditional Flow Matching.

I-CFM assumes an independent coupling between the source and the target, meaning that the joint distribution is simply the product of the marginals:

$$q_{I-CFM}(z) = q(x_0)q(x_1) \tag{28}$$

Practically, during the training, a sample x_0 from the source distribution is paired randomly with a sample x_1 drawn from the dataset. Given this coupling, the conditional probability path $p_t(x|z)$ is typically defined as a Gaussian distribution centered on the linear interpolation between x_0 and x_1. The mean of the path evolves linearly in time:

$$p_t(x|x_0, x_1) = \mathcal{N}(x|tx_1 + (1-t)x_0, \sigma^2) \tag{29}$$

Consequently, since we condition on both endpoints, we are no longer required to maintain a large variance to cover uncertainty of the starting gaussian distribution $\mathcal{N}(0, 1)$ such as in Lipman et al. [14] case. We can take the limit $\sigma \to 0$, collapsing the conditional probability as a Dirac delta centered on the two point of the starting and ending

distribution, enabling a deterministic path. In this deterministic limit, the expansion term vanishes, and the vector field simplifies to the pure time derivative of the linear interpolation path:

$$u_t(x|x_0, x_1) = x_1 - x_0 \tag{30}$$

This vector field instructs the flow to move points along straight lines connecting x_0 to x_1 with constant velocity.

Consequently, while methods based on Lipman's formulation are mathematically forced to use stochastic paths ($\sigma > 0$), I-CFM removes this constraint. This allows modern approaches (like Rectified Flow [8]) to employ fully deterministic conditional paths ($\sigma = 0$), leading to simpler regression targets, although a small σ can still be optionally used for regularization (fig. 13). We can thus model the conditional probability not as a diffuse Gaussian, but as a Dirac delta centered on a deterministic path $\psi_t(x_0, x_1)$. This derivation justifies the Independent Conditional Flow Matching (I-CFM) method, enabling the use of energy-minimizing straight lines ($x_1 - x_0$) that are more efficient than the path defined with the Lipman formulation.

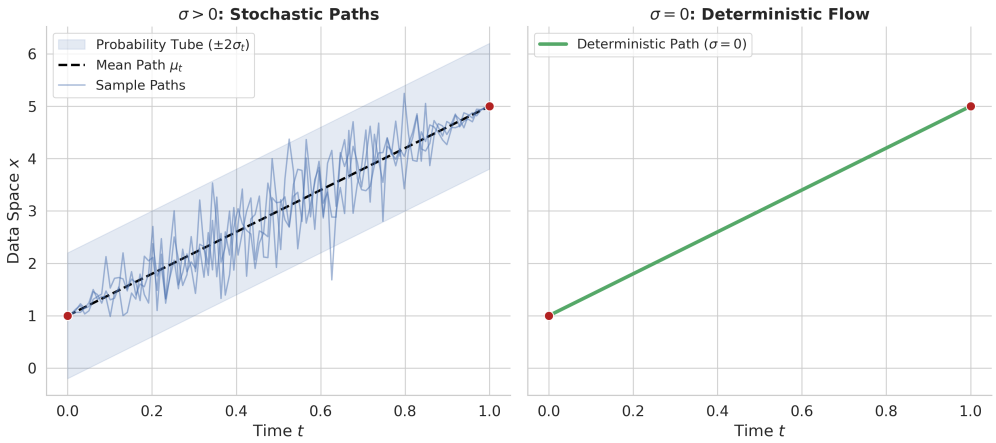

Figure 13: Left: The stochastic path formulation (characteristic of Lipman et al.'s original work) requires a non-zero width $\sigma > 0$ to avoid mathematical singularities. Individual sample trajectories fluctuate stochastically around the mean. **Right:** The generalized I-CFM framework allows for the limit $\sigma \to 0$, effectively collapsing the probability distribution onto a single, sharp deterministic trajectory.

2.7 Optimal Transport Conditional Flow Matching (OT-CFM)

Independent CFM successfully linearizes the conditional trajectories between source and target distribution, but from a global point of view, the entire process can be further optimized. In I-CFM the coupling from the source sample and target sample are paired randomly (independently), the resulting straight-line paths frequently cross and overlap in the probability space. This behaviour produces a marginal vector field $u_t(x)$, which represents the average velocity of all paths passing through x, obtaining a sub-optimal solution.

To overcome this inefficiency, Tong et al. [15] introduced the Optimal Transport Conditional Flow Matching (OT-CFM). The objective is to replace the random pairing with an optimized coupling that reduces the total transport cost between the starting and final distributions.

In this framework the mixing distribution $q(z)$ is no longer the product of the marginals but is defined by the optimal transport plan:

$$q(z) = \pi(x_0, x_1) \tag{31}$$

where $\pi(x_0, x_1)$ is the joint distribution that minimizes the Wasserstein cost between $q(x_0)$ and $q(x_1)$ (equation 21).

By coupling x_0 and x_1 according to the transport metric, OT-CFM enforces non-crossing straight paths (geodesics) also at the global level (fig. 14).

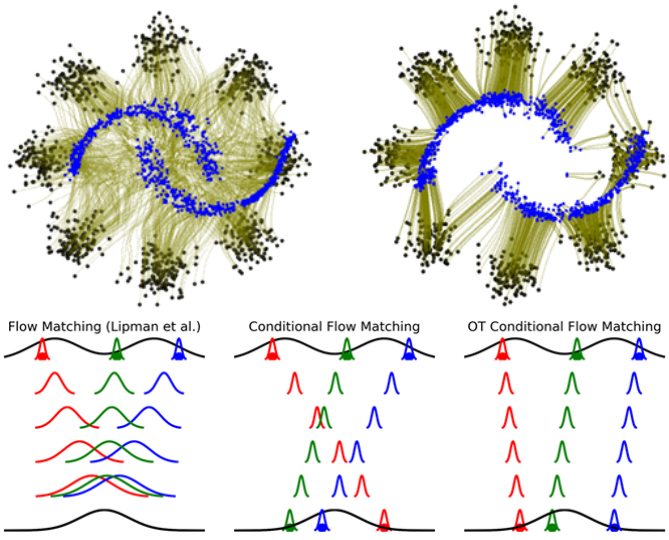

Figure 14: Top: Learned flows (green) from moons (blue) to 8gaussians (black) using I-CFM (left) and OT-CFM (right). Bottom: Left: Gaussian Conditional Flow Matching, Center: Independent Conditional Flow Matching, Right: Optimal Transport Conditional Flow Matching.

2.7.1 Minibatch Approximation

The computation of the exact transport plan π over the entire dataset introduces a significant computational bottleneck. Algorithms for solving the discrete Optimal Transport problem can be difficult to compute and store due to OT's cubic time and quadratic memory complexity in the number of samples, so we can proceed with the exact computation only for very small datasets.

To address this scalability issue, is possible to introduce the minibatch OT approximation. Instead of computing the global coupling, the algorithm samples a minibatch of size B from the source and target distributions at each training step. The optimal coupling is then solved locally within this (fig. 15). Tong et al. [15] empirically demonstrated the batch size can be much smaller than the full dataset size and still give good performance approximating the global trajectory and yield high-performance generative models.

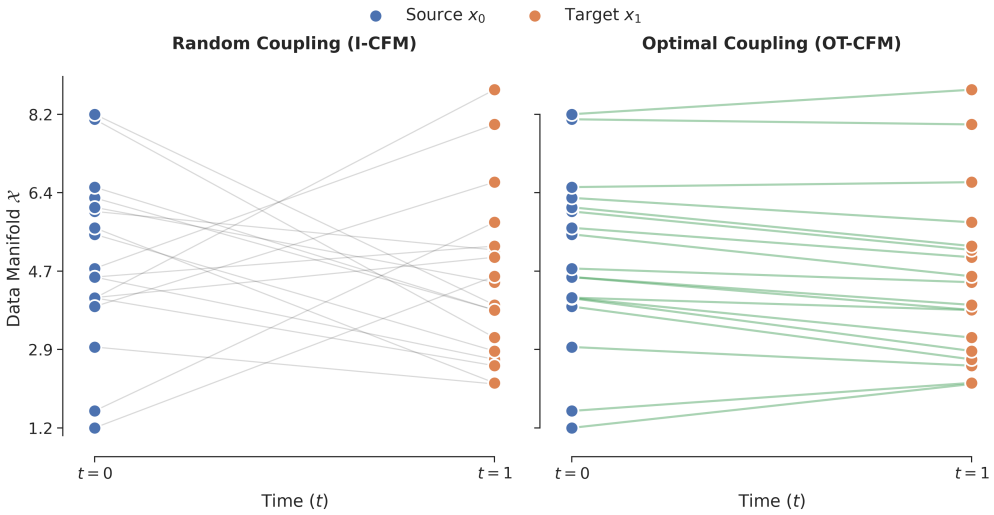

Figure 15: Left (I-CFM): Random independent coupling pairs source samples (x_0) and target samples (x_1) without geometric consideration. This results in chaotic, frequently intersecting paths ("crossing paths"), leading to a high-variance and complex marginal vector field. **Right (OT-CFM):** Optimal Transport coupling pairs samples to minimize the total displacement cost within the batch. This enforces monotonic, non-crossing trajectories (geodesics).

The main advantage in the introduction of the minibatch approximation is a significant reduction in complexity, passing from at least $\mathcal{O}(n^2)$ to $\mathcal{O}(kB^2)$ where B is the dimension of the batches and $k = n/B$ the total number of batches leading to $\mathcal{O}(kB^2) << \mathcal{O}(n^3)$ considering that the batch size B has upper bound n (batch size clearly can not be larger than the whole dataset).

Finally, it is insightful to observe the limit behavior of this approximation. In the extreme case where the batch size is set to $B = 1$, the "optimal" coupling within the batch is trivial (pairing the single source with the single target). Since batches are sampled randomly, this setting effectively collapses back to the Independent CFM (I-CFM) formulation described previously.

To conclude this chapter, we summarize in a table 2 the main characteristics of the most used Flow Matching methods.

Probability Path	q_z	$\mu_t(z)$	σ_t	Cond. OT	Marg. OT	Gen. src
Flow Matching [14]	$q(x_1)$	tx_1	$1-(1-\sigma)t$	✓		
I-CFM [15]	$q(x_0)q(x_1)$	$tx_1+(1-t)x_0$	σ	✓		✓
Rectified Flow [8]	$q(x_0)q(x_1)$	$tx_1+(1-t)x_0$	0	✓		✓
OT-CFM [15]	$\pi(x_0,x_1)$	$tx_1+(1-t)x_0$	σ	✓	✓	✓
SB-CFM [15]	$\pi(x_0,x_1)$	$tx_1+(1-t)x_0$	$\sigma\sqrt{t(1-t)}$	✓	✓	✓

Table 2: Comparison of different Conditional Flow Matching formulations. The table outlines the choices for the mixing distribution $q(z)$, the conditional path parameters (μ_t, σ_t), and highlights which methods support generalized source distributions (Gen. Src) and minimize Optimal Transport costs locally (Cond. OT) or globally (Marg. OT).

3 MAISI framework

The Medical AI for Synthetic Imaging (MAISI), developed by NVIDIA, represents a state-of-the-art framework for generating synthetic 3D total body computed tomography (CT) volumes. The main innovation of MAISI lies in its ability to generate high-resolution volume (up to $512 \times 512 \times 768$ voxels) while maintaining anatomical plausibility through semantic control (Fig. 16).

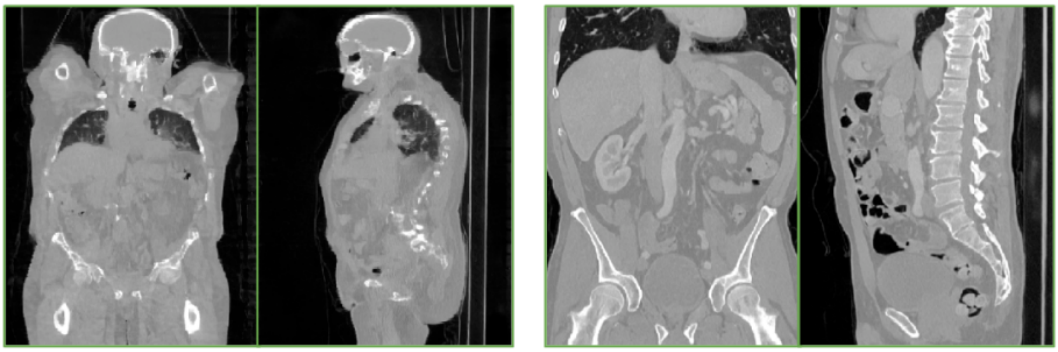

Figure 16: Examples of generated MAISI images with different region inputs.

The framework addresses the computational intractability of generating 3D data into pixel space leveraging a Latent diffusion model (LDM) architecture. The model encodes high-dimensional CT volumes into a compressed latent space, in this way we have a lighter version of the volume that is computationally tractable to work with. Furthermore, by incorporating a ControlNet mechanism, MAISI can condition the generation process on specific organ segmentation masks, enabling the synthesis of paired image-label datasets essential for downstream supervised learning tasks.

3.1 MAISI architecture

As shown in (fig. 17), the MAISI pipeline operates in three distinct stages: volume compression through a Variational Autoencoder, latent generation through a diffusion-based model, and semantic conditioning through ControlNet.

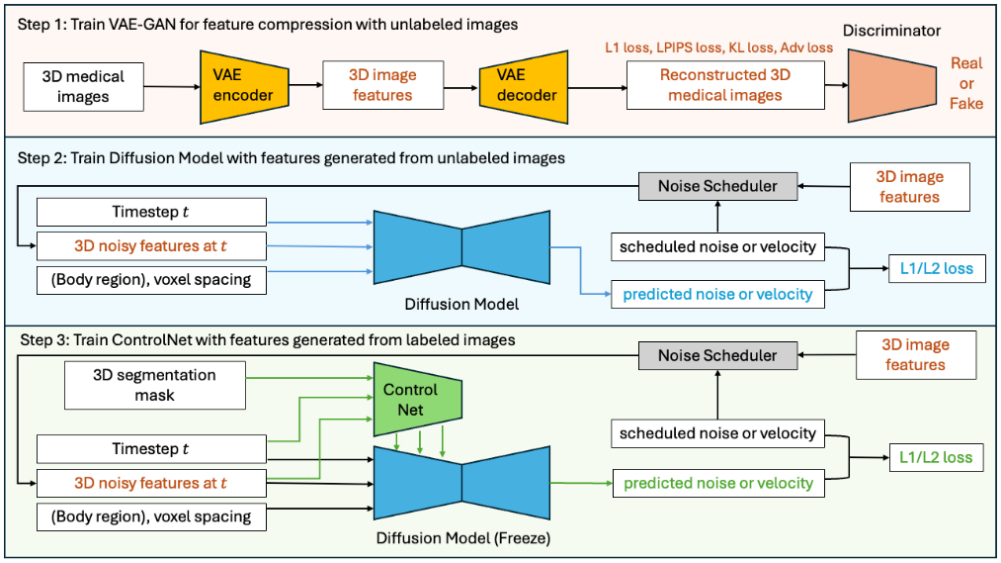

Figure 17: MAISI train scheme [6]

3.1.1 Stage 1: Foundation Volume Compression Network (VAE)

The first stage involves a 3D Variational Autoencoder (VAE), referred to as the Foundation Volume Compression Network. This component is trained to compress the input volume into a lower-dimensional latent representation while preserving essential anatomical details.

The Volume Compression Network (MAISI VAE) is trained on a dataset comprising 37243 CT volumes for training and 1963 CT volumes for validation, covering the chest, abdomen, and head and neck regions. To establish the VAE as a foundational model, an extensive range of data augmentation techniques is employed. For CT images, intensities are clipped to a Hounsfield Unit (HU) range of -1000 to 1000 and normalized to a range of [0,1] and images underwent spatial augmentations, such as random flipping, random rotation, random intensity scaling, random intensity shifting, and random upsampling or downsampling.

Given a grayscale CT volume $x \in \mathbb{R}^{C \times H \times W \times D}$ in voxel space, where C denotes the channels (equal to 1 in the gray level pixel space), H denotes the height, W the width and D the depth. The encoder $\mathcal{E}$ of the MAISI-VAE downsamples x to a latent representation $z \in \mathbb{R}^{4C \times H/4 \times W/4 \times D/4}$, so we have a reduction of the total voxel elements of

a factor 16. A decoder $\mathcal{D}$ reconstructs the volume starting from the latent variable z, so we have the reconstructed volume $\tilde{x} = \mathcal{D}(z) = \mathcal{D}(\mathcal{E}(x))$. A 3D discriminator, denoted as $\mathcal{C}$, is utilized to identify and penalize any unrealistic artifacts in the reconstructed volume $\tilde{x}$. The overall objective $\mathcal{L}_{AE}$ used to train the volume compression network $(\mathcal{E}, \mathcal{D})$ in MAISI can be defined as follows:

$$min_{E,D}max_{C}(\mathcal{L}_{recon}(x, \mathcal{D}(\mathcal{E}(x))) + \mathcal{L}_{lpips}(x, \mathcal{D}(\mathcal{E}(x))) + L_{reg}(\mathcal{E}(x)) + L_{adv}$$

where

$$L_{adv} = log\mathcal{C}(x) + log(1 - \mathcal{C}(\mathcal{D}(\mathcal{E}(x)))) \tag{32}$$

The MAISI VAE model is trained with 8 32G V100 GPU with four different type of losses (reconstruction loss, perceptual loss (LPIPS), Kullback-Leibler (KL) divergence regularization and adversarial loss), initially trained for 100 epochs using randomly cropped patches of size [64,64,64]. This approach is adopted to improve the model's ability to generalize to images with partial volume effects. After this initial phase, training is continued for an additional 200 epochs using larger patches of size [128,128,128], which allows the model to capture more contextual information and improve overall accuracy. The MAISI VAE is used to compress the latent features that will be employed in latent diffusion models, where having a well-structured and meaningful latent space is crucial for effective diffusion dynamics. Therefore, during MAISI VAE training, the weight of the KL loss is adjusted to ensure the standard deviation remains between 0.9 and 1.1. This calibration balances the model's focus between accurate data reconstruction and adherence to the prior distribution. As the MAISI VAE is intended to serve as a foundational model, maintaining this balance also helps to prevent over-fitting [6].

3.1.2 Stage 2: Latent diffusion model (LDM)

The core generative engine of MAISI operates within the compressed latent space produced by the VAE. The design of this component marks the difference between the two version of the framework. The initial version of MAISI [6] employed a standard Latent Diffusion Model (LDM) based on Denoising Diffusion Probabilistic Models (DDPM) [19]. It utilized a U-Net backbone to iteratively denoised the latent representation, while effective, this approach suffered from the inherent slow inference speed of DDPMs, requiring hundreds of function evaluations (NFE) to generate a single volume.

To improve the computational efficiency, especially during the inference, the second version of MAISI framework [20] replaces the traditional DDPM formulation with a Flow Matching method called Rectified Flow [8], With this transition, MAISI represents the current state-of-the-art in high-resolution 3D volume generation.

The final training set consists of over 107k images generated from the 11k scans. Data preprocessing for diffusion model training involves applying a series of precise

transformations to the image data, including loading the images, ensuring the correct channel structure, adjusting the orientation according to the ”RAS” axcode, and scaling intensity values from -1000 to 1000 to normalize the data between 0 and 1. The process further refines the images by adjusting dimensions to the nearest multiple of 128 to ensure a better handling by the U-Net architecture, using trilinear interpolation. Then each image is passed through the pre-trained autoencoder, generating a compressed latent representation that is saved for subsequent model training.

It's developed a three-stage training strategy with mixed-precision optimization. The only conditioning signal is the voxel spacing, which is automatically extracted from data and requires no manual annotation. Training of MAISI v2 is very complex an require a big amount of computational recources, is conducted on a cluster of 64 NVIDIA A100 GPUs (80GB) over three weeks using the AdamW optimizer.

Pre-training Stage: In the first stage, MAISI use low resolution images of size 128 × 128 × 128, consisting of both naturally low-resolution volumes and high-resolution images that have been downsampled. Since all images share the same shape, is possible to use a large batch size of 96 on 80GB A100 GPUs. This stage is critical as it not only accelerates training significantly but also helps prevent NaN issues that may arise in subsequent stages. Is used a learning rate of 1e-3 to train this stage for one day.

Main Stage: Once the low-resolution model converges, the second stage continue training on the full-resolution dataset, where images vary in spatial dimensions. To avoid using batch size as 1, is used bucketed data parallelism. Images are grouped by shape and distribute them to separate GPUs, ensuring that each GPU processes only images of the same shape, and then train the model with distributed data parallel. This allows to use larger batch sizes where possible. For example, 128 × 128 × 128 images are trained with a batch size of 96, 256×256×128 with a batch size of 24, and 512×512×768 with a batch size of 1. Is used a learning rate of 1e-3 and polynomial decay for this stage for 16000 epochs. It took around 10 days. While this strategy greatly accelerates training, it introduces data imbalance issue.

Fine-Tuning Stage: To address the data imbalance issue from the second stage, is added a third fine-tuning stage, where the model is trained using all images mixed together with a fixed batch size of 1 and sampling weights to balance the contribution from different datasets. This ensures the model is exposed to the full data distribution and helps reduce any size-related bias in the final representation. Is used a learning rate of 1e-4 for this stage for around 2000 epochs. It took around 10 days.

3.1.3 Stage 3: Conditioned generation (ControlNet)

In medical imaging, unconstrained generation is of limited utility. To enable the generation of specific anatomies (e.g., a liver with a tumor at a precise location), MAISI integrates ControlNet [21]. ControlNet creates a trainable copy of the diffusion model's encoding layers. These trainable layers take segmentation masks (covering up to 127

anatomical structures) as input and inject this spatial information into the frozen unconditioned main latent model trained in the previous step. This allows MAISI to generate CT volumes that perfectly align with a user-provided segmentation mask, bridging the gap between generative AI and supervised segmentation tasks.

3.2 Rectified Flow vs OT-CFM

Despite the state-of-the-art results achieved by the MAISI framework, training a Rectified Flow model on high-dimensional data remains resource intensive, in this section we analyze the Rectified Flow method utilized by MAISI [20][8] and compare it with the proposed OT-CFM approach [15].

Rectified flow is a Flow Matching method where the main objective is to learn a marginal vector field $u_t(x)$ that is as straight as possible. The aim of this process is to enable fast inference by solving the Neural ODE with a single Euler step ($NFE = 1$).

The foundational logic is grounded on the principles of the Independent-CFM (I-CFM) discussed in chapter 2. However, simply training a model between the starting distribution $q(x_0)$ and the final distribution $q(x_1)$ (also called as 1-Rectified Flow) is often insufficient for one-step generation. To address this, the method proceeds with a recursive step involving the generation of synthetic samples $\hat{x}_1$ by solving the ODE:

$$\hat{x}_1 = \text{ODE\_Solve}(v_t^1, x_0, t \in [0, 1]) \tag{33}$$

This creates a new dataset of pairs $(x_0, \hat{x}_1)$ where $\hat{x}_1$ is casually connected to x_0 by the flow of the first model. This coupling allows for learning significantly straighter flows between the source and the new target distribution.

It's crucial to note that with this method we lose the simulation-free framework because the ODE must be explicitly solved for the entire dataset N-1 times to generate the synthetic targets required for the N-Rectified Flow training.

In figure 18 we have a useful example to visualize the vector field v_t^1 trained on randomly sample pairs (x_0, x_1). Even if the model is able to learn a velocity field that transport $q(x_0)$ to $q(x_1)$, the resulting trajectories are not perfectly straight. While they approximate linear interpolation globally, they exhibit significant curvature (or "rewiring") in high-density regions where paths intersect. Due to these non-straight, "V"-shaped paths, solving the ODE with a single step (N=1) yields poor performance, failing to accurately reconstruct the target distribution.

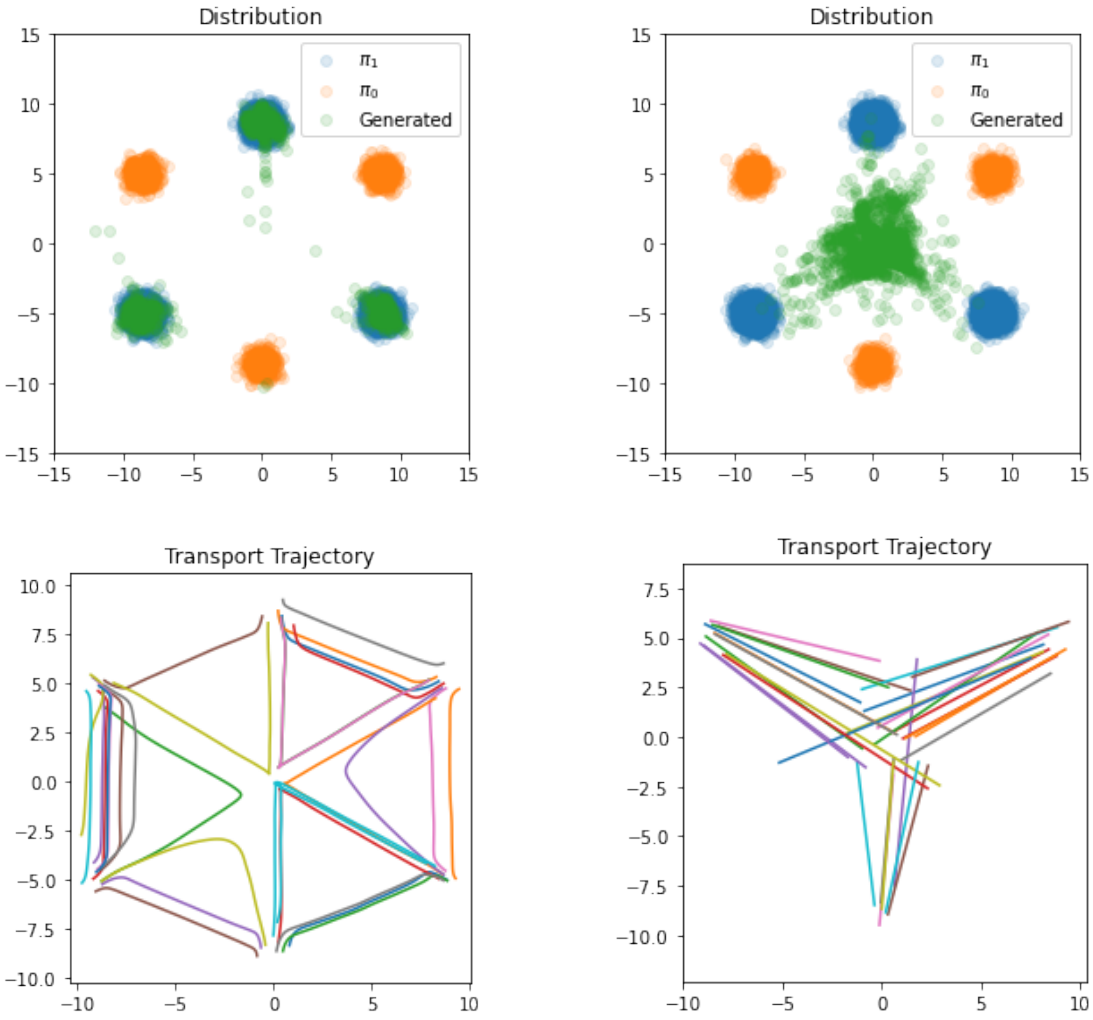

Figure 18: 1-Rectified Flow Trajectories. Left ($N = 100$): "V-shape" trajectories are present, but generations are good. **Right** ($N = 1$): There are very few trajectories that goes to the target distribution, generations are very bad. Image from [7].

To overcome the curvature issue, in fig 19 is employed a Reflow procedure to obtain a straightened flow, denoted as 2-Rectified Flow. This involves repeating the training procedure, but replacing the random pairs (x_0, x_1) with the flow coupled pairs $(x_0, \hat{x}_1)$, where $\hat{x}_1$ is the synthetic sample generated from the simulation of the 1-Rectified Flow, so the new linear interpolation becomes:

$$x_t = t\hat{x}_1 + (1 - t)x_0. \tag{34}$$

the 2-Rectified Flow learns a velocity field that transports the starting distribution to the synthetic one with nearly perfectly straight trajectories.

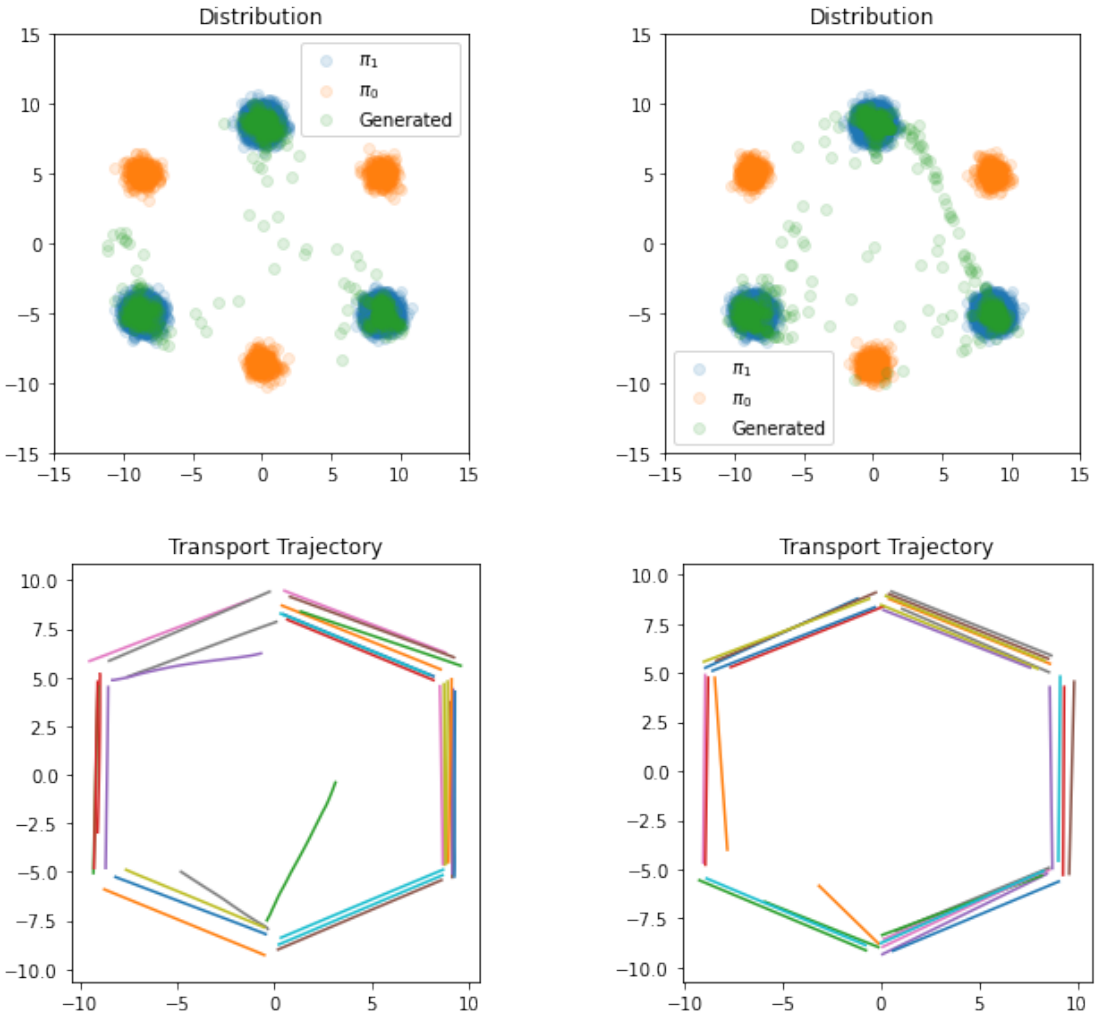

Figure 19: 2-Rectified Flow Trajectories. Left ($N = 100$): The trajectories are now straightened, connecting source and target with constant velocity. **Right** ($N = 1$): Thanks to the rectification, the single-step generation yields results almost identical to the multi-step solver. Image from [7].

The key result is that the transport trajectory has been effectively straightened. Consequently, we maintain high accuracy even when solving the ODE with a single Euler step (N = 1) (fig. 20).

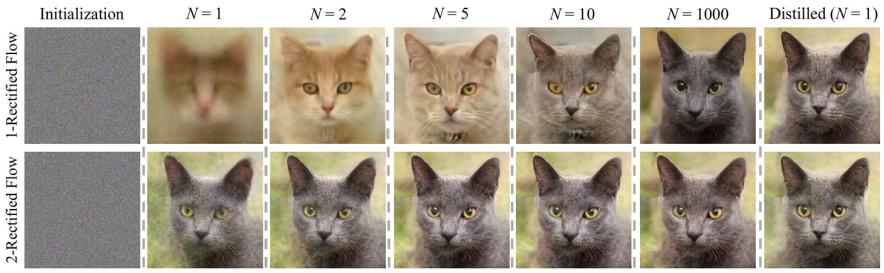

Figure 20: Difference between different step integration between 1-Rectified flow and 2-Rectified flow. Image from [8].

This is true for simple distribution, in more complex, high-dimensional cases, to rectify the flows is ideally needed infinite reflow iteration to ensure that every path are perfectly straight. However, empirically, the effectiveness of the reflow procedure is highest in the first iteration and exponentially decrease with successive iterations, thus, even in most complex cases, one or very few Reflow iteration are typically sufficient to achieve good generation with one or very few NFE.

A critical trade-off to consider is error propagation. Since the Reflow procedure uses a synthetic dataset generated by the previous model as its target, the 1-Rectified Flow effectively acts as an upper bound on performance. If the initial model suffers from mode collapse or other generation errors, the 2-Rectified Flow (and subsequent iterations) will learn and potentially amplify these errors.

3.3 MONAI environment

The implementation of the MAISI framework and the proposed OT-CFM optimizations rely on the Medical Open Network for AI (MONAI) [22]. MONAI is an open source [23], PyTorch based framework specifically designed to work on challenges presented in medical imaging, such as high-dimensional data handling and reproducibility. This ecosystem provides specialized building blocks used in medical imaging such as the modules used in generative pipeline described above.

Processing 3D CT volumes requires specialized I/O operations and geometric transformations that are not natively supported by standard computer vision libraries. The framework utilizes specific modules to ensure data consistency:

- **Data I/O and Orientation:** The `monai.data` module handles the loading of files and parses the header information (affine matrix, voxel spacing). This is critical for extracting the physical resolution used to condition the generative model. Furthermore, transforms ensure all volumes are oriented to a standard coordinate system (e.g., RAS: Right-Anterior-Superior) before processing.
- **Patch-based Pipeline:** Given the GPU memory constraints of processing 512^3 volumes, MONAI's transformation pipeline manages the extraction of random spatial patches during training and the reassembly of patches during inference, abstracting the complexity of tensor slicing and padding.

The architecture of MAISI is built upon the `monai.generative` package, which provides standardized implementations of the core neural networks.

- **Architectural Modularity:** The Foundation VAE and the diffusion backbone are instances of the `AutoencoderKL` and `DiffusionModelUNet` MONAI classes [23], respectively. These implementations include 3D-specific layers (such as 3D convolutions and attention mechanisms) optimized for volumetric data.
- **Sliding Window Inference:** A critical component for the generation phase is the `SlidingWindowInferer`. Since the model is trained on patches (e.g., 128^3), generating a full body volume requires predicting multiple overlapping patches and blending them seamlessly to avoid boundary artifacts.

The modular design of the MONAI ecosystem allows for the separate usage of the network architectures. This feature is very useful in the context of this thesis, it allows us to retain the pre-trained MAISI-VAE as a frozen feature compressor, allowing us to focus our computational resources entirely on training the generative U-Net with the novel OT-CFM solver.

4 Implementation

4.1 Dataset: CT-Rate

The CT-RATE (Computed Tomography Radiology Text and Evaluation) dataset represents an open-source resource developed to address the data limitation in artificial intelligence applied to 3D medical imaging, is a large-scale, publicly available dataset that pairs 3D medical images with their corresponding textual reports, each volume is annotated with 18 abnormalities [24].

The total dataset includes 50,188 reconstructed 3D CT volumes, equivalent to over 14.3 million 2D slices, originating from 21,304 unique patients. Each volume is reconstructed using different techniques to observe different lungs details, data comes from the Istanbul Medipol University Mega Hospital and was acquired between May 2015 and January 2023. The reports were anonymized and translated from Turkish to English before being included. The dataset also includes organ-level anatomical segmentation labels generated using an advanced segmentation model. These labels have not been clinically validated.

Despite being a single-center dataset, CT-RATE reflects clinical reality by presenting notable heterogeneity, which is crucial for the development of robust models. The variety of CT volumes stems from the use of scanners from three different manufacturers (Philips 61.5%, Siemens 30.1%, PNMS 8.4%). Volumes are reconstructed using different techniques (sharper kernels for lungs, smoother for mediastinum) and exhibit variations in slice thickness, axial pixel spacing, and the number of slices.

4.1.1 Stratification

Due to available computational resources we selected 2250 volumes to download using stratified sampling based on age and sex. This approach ensures that the distribution of the selected subset preserves the characteristics of the original population. The size of the selected subgroup dataset is about 500 GB.

- Sex: Male (M), Female (F)
- Age groups: 18-30, 31-40, 41-50, 51-60, 61-70, 71-80, 81-90

4.1.2 Pre-processing

The first step of the analysis is to prepare the dataset for the neural network. Since the raw NifTi files (.nii.gz) comes from different detectors, a harmonization pipeline was applied before feeding the network:

- **Intensity Normalization:** Hounsfield Unit (HU) intensities clipped to the range [-1000, 1000] to suppress outliers and normalize the dynamical range.
- **Resampling:** Volumes resampled to a fixed physical spacing of [0.684, 0.684, 2.422] mm using B-spline interpolation.
- **Spatial Cropping:** Voxel grids were cropped or padded to a uniform resolution of 256 x 256 x 128 according to available computational resources.

4.2 Encoding: MAISI-VAE

We used the pretrained VAE-GAN of the MAISI framework to encode our dataset into the latent space.

When we attempted to fine-tune the model to improve the reconstruction quality, we observe that the balance of the loss is very subtle, in few epochs the model shows instability, leading to loss explosion and a degradation of the reconstructed images. To mitigate this, we drastically reduced the learning rate ($\sim 10^{-}7$) to maintain the stability of the model. However, with this settings, we can observe that the loss oscillates around the same pretrained values, indicating that the pre-trained weights were already near optimal for our data distribution.

We have this behaviour for different reasons, first of all the domain of the problem is highly specific, so the VAE demonstrates a robust capability to reconstruct even if our dataset contains out-of-distribution images. Second, this robustness comes from the extensive pre-training conducted by Maisi team on a large-scale dataset comprising over 39,000 CT and 18,000 MRI volumes, trained for 300 epochs using 8 NVIDIA V100 (32GB) GPUs. So in the end we decided to proceed with the encoding of our volumes using the pre-trained weights of the model, omitting the fine-tuning stage.

This procedure is aligned with the methodology validated in the original MAISI literature. In the first version of Maisi paper [6], they show that the performances of their own Maisi-VAE achieve a performance comparable to dedicated models on unseen datasets [fig. 21] without the need of additional training.

Dataset	Model	LPIPS ↓	SSIM ↑	PSNR ↑	GPU ↓
MSD Task07	MAISI VAE	**0.038**	**0.978**	**37.266**	**0h**
	Dedicated VAE	0.047	0.971	34.750	619h
MSD Task08	MAISI VAE	0.046	0.970	36.559	**0h**
	Dedicated VAE	**0.041**	**0.973**	**37.110**	669h
Brats18	MAISI VAE	**0.026**	**0.0977**	**39.003**	**0h**
	Dedicated VAE	0.030	0.0975	38.971	672h

Figure 21: Performance comparison of the MAISI VAE model on out-of-distribution datasets versus dedicated VAE models. The "GPU" column shows additional GPU hours for training with one 32G V100 GPU[6].

4.3 Diffusion Model Architecture: U-Net

The core of the generative framework is the neural network tasked with learning the conditional vector field $v_t(x, t)$. Since the diffusion process operates within the compressed latent space produced by the VAE, the chosen architecture is a **3D U-Net** optimized for volumetric data with lower spatial resolution but high semantic content.

For implementation, we adopted the `DiffusionModelUNet` class provided by the MONAI framework, which offers specialized building blocks for medical imaging and ensures native integration with 3D tensors [23].

4.3.1 Network Architecture

Unlike standard U-Nets typically used for segmentation tasks, this variant is specifically designed for denoising (or *velocity prediction* in the context of flow matching). The network takes as input the latent volume $x_t \in \mathbb{R}^{4\times64\times64\times32}$ (as described in general in chapter 3.1.1 and the temporal information t, returning an estimate of the velocity field v_t with the same dimensionality as the input.

The specific configuration adopted in our experiments was calibrated to balance the model's expressive capacity with computational resource constraints (single GPU training). The main architectural parameters are as follows:

- **Spatial Dimensionality:** The model operates with 3D latent volumes of size [4, 64, 64, 32]. This resolution results from the compression factor of the VAE applied to the original input volumes.
- **Channels:** The network features a symmetric encoder-decoder structure with three resolution levels. The feature map channels for each level are set to **(64, 128, 256)**, respectively.
- **Residual Blocks:** Each level of the hierarchy processes features through **2 consecutive residual blocks**, ensuring sufficient depth to learn complex abstractions without incurring the vanishing gradient problem.
- **Attention Mechanisms:** Given the high computational cost of the *self-attention* operation in 3D (which scales as $\mathcal{O}(N^2)$), we applied attention blocks **exclusively at the deepest level** (the bottleneck), where the spatial resolution is minimal ($16\times16\times8$). The upper levels (64 and 128 channels) utilize standard convolutions, significantly reducing VRAM usage while maintaining receptive field integrity.
- **Time Embedding:** Temporal conditioning is handled via a sinusoidal *Time Embedding* projected into a dense vector, which is injected into each residual block through a *scale-and-shift* mechanism, allowing the network to modulate its response based on the time step t of the flow trajectory.

```
from monai.networks.nets import DiffusionModelUNet

unet = DiffusionModelUNet(
    spatial_dims = 3,
    in_channels = 4,
    out_channels = 4,
    channels= [64, 128, 256],
    attention_levels = [False, False, True],
    num_res_blocks = 2,
    num_head_channels = 32,
)
```

The training of the U-Net was performed by minimizing the *Flow Matching Loss* (MSE between the predicted velocity and the target vector field u_t). To ensure numerical stability and maximize throughput on the available hardware, we adopted the following implementation strategies:

- **Optimizer:** We employed the **AdamW** optimizer with an initial learning rate of 1×10^{-3} and a weight decay of 1×10^{-4} to regularize weights and prevent overfitting.
- **Learning Rate Scheduler:** To facilitate convergence towards robust local minima, the learning rate is modulated using a **Cosine Annealing** strategy, which progressively reduces the step size following a cosine curve down to a minimum value of 4×10^{-5}.
- **Numerical Precision:** The entire training loop was implemented using **bfloat16** (Brain Floating Point) precision. Unlike standard `float16`, `bfloat16` maintains the same dynamic range (exponent bits) as `float32`. This prevents the underflow issues typical in diffusion models while halving memory usage and leveraging the specialized tensor cores of modern GPUs.
- **Normalization**: The latent vectors are normalized as $\mathbf{z}_{\text{norm}} = \frac{\mathbf{z}-\mu}{\sigma+\epsilon}$, where $\mu = -0.082876$ and $\sigma = 0.733372$.

```
import torch

optimizer = torch.optim.AdamW(unet.parameters(), lr=
   learning_rate, fused=True, weight_decay = 1e-5)

scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(
    optimizer,
    T_max = num_epochs,
```

```
    eta_min = 4e-5
)
```

4.3.2 Training Strategy: Decoupling OT and Backpropagation Batch Sizes

A critical point in implementing Optimal Transport Conditional Flow Matching (OT-CFM) lies in the conflict between theoretical requirements and hardware constraints. As discussed in Section 2.7.1, the Minibatch OT approximation relies on solving the optimal transport problem within a sampled batch of data. The quality of this approximation and consequently the straightness of the learned trajectories depend on the batch size B. A small batch size (e.g., $B = 1$) degenerates the OT coupling into a random independent coupling (I-CFM), effectively nullifying the benefits of the geometric optimization.

However, processing large batches of 3D latent volumes is memory intensive, Even in a compressed latent space, loading a large batch (e.g., $B = 128$) alongside the U-Net parameters and the activation maps required for backpropagation needs hundreds of GB available in the VRAM, so we need to find a strategy to split this different tasks.

To address this bottleneck, we implemented a **Gradient Accumulation** strategy that decouples the batch size required for the OT statistics from the batch size necessary for the backpropagation required to compute the loss (limited by GPU memory).

In our implementation, the training loop is structured as follows:

1. **Macro-Batch Sampling:** The DataLoader load a large batch (Macro-Batch) of size $N = 128$ from the dataset. This size ensures a good statistical minibatch, sufficient to obtain a good approximation of the global distribution. This is the stage where the Optimal Transport coupling is performed, pairing source samples x_0 and target samples x_1 to minimize the transport cost within the set of N samples.

2. **Micro-Batch Processing:** The macro-batch is then sliced into smaller chunks (Micro-Batches) of size $M = 16$, which fits within the GPU memory allowing the evaluation and update of the neural network.

The network processes the micro-batches sequentially without updating the weights immediately. For each micro-batch k, the loss $\mathcal{L}_k$ is computed and scaled by the ratio of the micro-batch size to the macro-batch size:

$$\mathcal{L}_{norm} = \mathcal{L}_k \times \frac{M}{N} \tag{35}$$

The gradients are calculated via backpropagation and accumulated in the model's parameters. The optimizer step is performed only after the entire macro-batch has been processed (i.e., after N/M iterations).

This approach allows us to train the model with an effective batch size of 128, enabling the geometric benefits of OT-CFM, while maintaining a peak memory footprint equivalent to a batch size of 16. The implementation details are summarized in the pseudo-code below:

```
# Pseudo-code of the implemented training loop

from torchcfm.conditional_flow_matching import
   ExactOptimalTransportConditionalFlowMatcher

macro_batch_size = 128
micro_batch_size = 16

# ...

for epoch in num_epochs:
    unet.train()
    train_epoch_loss = 0.0

    for macro_batch in dataloader:
        optimizer.zero_grad()

        # ...

        # Compute optimal transport between the couple
        t, x_t, u_t = fm.sample_location_and_conditional_flow(
            x_0, x_1, t=t
        )

        # ...

        # Iterate over micro-batches
        for i in range(0, macro_batch_size, micro_batch_size):
            # Slice the tensors
            xt_micro = xt[i : i+micro_batch_size]
            ut_micro = ut[i : i+micro_batch_size]
            t_micro  = t [i : i+micro_batch_size]

            # Forward pass
            v_pred = unet(xt_micro, t_micro)

            # Compute Loss
            loss = torch.mean((v_pred - ut_micro) ** 2)
```

```
            # Scale loss for correct mean calculation
            loss_normalized = loss * (micro_batch_size /
               macro_batch_size)

            # Accumulate gradients
            loss_normalized.backward()

        # Update weights once per macro-batch
        torch.nn.utils.clip_grad_norm_(unet.parameters(),
           max_norm=1.0)
        optimizer.step()

        # ...

scheduler.step()
```

4.4 Generation

Once the model is trained, we can generate new synthetic latent representations through the integration of an Ordinary Differential Equation (ODE). The generative process starts by sampling a random latent vector x_0 from the starting distribution (standard Gaussian noise), and evolving it according to the vector field learned by the U-Net architecture over the time interval $t \in [0, 1]$.

In according to do that we use the `torchdiffeq` library which provides different ODE solvers: the Flow Matching objective encourages the learning of straight trajectories (which theoretically allows for simple Euler integration), employing higher-order solvers such as the fourth-order Runge-Kutta (rk4) or adaptive solvers like dopri5 ensures higher fidelity in the reconstructed distribution, minimizing discretization errors along the path. For fixed step solvers (like Euler or RK4), is possible to tune the Number of Function Evaluations (NFE) via the step size, whereas for adaptive solvers, is possible to tune the error tolerance (atol, rtol) that the solver must satisfy.

```
from torchdiffeq import odeint

def run_inference(model, latent_shape, condition, device,
   method, steps, x0):

    # ...

    # ODE solver integration
    with torch.no_grad():
        traj = odeint(
            node,
            x0,
            t_span,
            method = method,
            atol = 1e-5,
            rtol = 1e-5
        )

    # Last generated point of the trajectory is the generated
       data
    generated_latent = traj[-1]

    return generated_latent

# ...

# execution example
```

```
latents_denormalized = run_inference(
    model = unet,
    x0 = None,
    latent_shape = [1, 4, 64, 64, 32],
    condition = None,
    device = 'cuda',
    method = 'rk4',
    steps = 40
)


# Inverse normalization (using dataset statistics)
latents_out = latents_denormalized * (global_std + 1e-8) +
   global_mean
```

4.5 Decoding

The results of the generation process is a tensor that live in the compressed latent space $\mathbb{R}^{4\times64\times64\times32}$. This representation contains the semantic of the generated volume but it is not yet a viewable image. To return to the original voxel space, the latent representation must pass through the decoder module of the VAE.

```
with torch.inference_mode():
    with torch.autocast(device_type="cuda", dtype=torch.
       bfloat16):
        x = autoencoder.decode(z)

        # Cast to float32 because numpy doesn't support
           bfloat16
        x = x.float()

x = x.cpu().detach().numpy()
```

The decoding step involves a significant spatial upsampling. To manage memory efficiency and ensure numerical stability, we perform this operation using mixed precision (bfloat16), casting the result back to standard float32 only for the final visualization or storage.

5 Results

5.1 Training Dynamics and Convergence Analysis

We train three different models using the identical architectural structure described in Chapter 4 only changing the objective function to reflect different probability paths: Optimal Transport Conditional Flow Matching (OT-CFM), Independent CFM (I-CFM, equivalent to 1-Rectified Flow), and Variance-Preserving CFM (VP-CFM), which emulates a diffusion schedule. Figure 22 illustrates the training loss trends for each method. To obtain an accurate comparison between the convergence dynamics, we must account for the intrinsic differences in the magnitude and complexity of their respective target vector fields. Specifically, the VP-CFM path relies on trigonometric interpolants with a higher expected squared norm, resulting in a structurally higher Mean Squared Error (MSE) compared to the constant, linear targets of I-CFM and OT-CFM.

The learning curves demonstrate that all models successfully converge to a stable minimum. A marginal difference is notable between I-CFM and OT-CFM, which is attributable to the pairing method between the source and target distributions. However, both linear formulations achieve lower losses than rescaled VP-CFM, confirming that the path designed by the diffusion schedule is geometrically suboptimal compared to the optimal transport interpolant as described in Chapter 2.5.2.

Furthermore, a detailed inspection of the initial training epochs reveals distinct convergence behaviors among the three models. OT-CFM is the only formulation that does not exhibit a noisy descent or sudden loss spikes during the early stages of optimization. This early stability empirically demonstrates that the Optimal Transport coupling significantly simplifies the learning objective for the neural network by providing a globally ordered, non-intersecting vector field from the very first iterations.

Additionally, VP-CFM displays a noisier loss curve even during the advanced, stable phases of training. This persistent variance further highlights the inherently complex nature of the curved diffusion schedules compared to the more efficient linear interpolants used in I-CFM and OT-CFM.

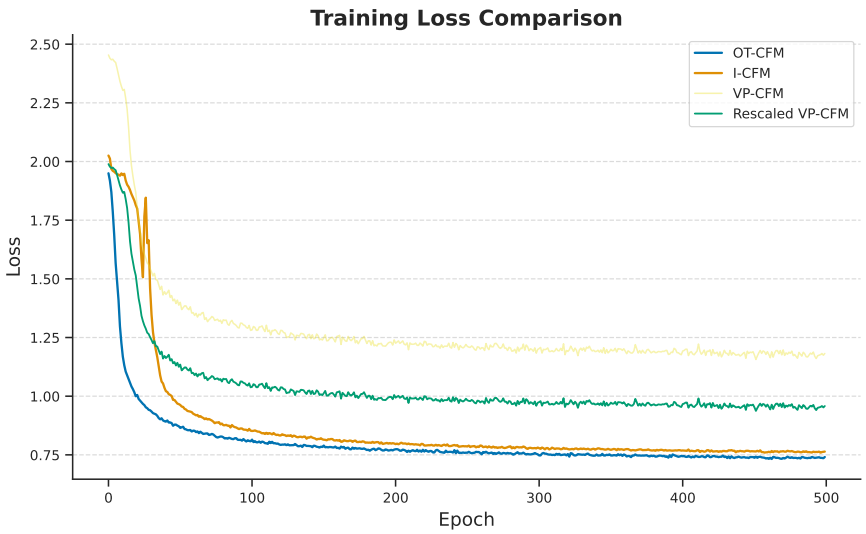

Figure 22: Training loss trends for OT-CFM, I-CFM, and VP-CFM models. Note that the raw Mean Squared Error (MSE) of the VP-CFM model has been analytically rescaled by a factor of 1.233 (the ratio of the expected squared norms of their respective target vector fields). This normalization accounts for the intrinsically higher magnitude of the VP-CFM trigonometric target, allowing a coherent visual comparison of the convergence dynamics alongside the linear paths of I-CFM and OT-CFM. The minimum losses reached: OT-CFM = 0.719, I-CFM = 0.759, Rescaled VP-CFM = 0.937.

5.2 Inference Efficiency and Qualitative Assessment

Following the train phase, we proceed with the generation of 1000 latent samples with each trained model, we used the adaptive step integrator "dopri5", which dynamically adjusts the Number of Function Evaluations (NFE) based on the vector field complexity. As shown in table 3 I-CFM and OT-CFM achieve a comparable inference time, both of which are faster than VP-CFM that follows diffusion schedule. This confirm the theoretical better efficiency of the linear interpolant over the diffusion schedules.

	I-CFM	OT-CFM	VP-CFM
Time (min)	129.2 min	132.7 min	161.6 min

Table 3: Inference time for the generation of 1000 samples.

To establish a baseline for the generative capabilities of our framework, we first show examples of successful volumetric synthesis. Figure 23 demonstrates the high-fidelity reconstruction of the skeletal structure from a generated latent volume, rendered using 3D Slicer. This highlights the model's ability to preserve complex, high-frequency 3D morphological features. Similarly, Figure 24 presents a cross-sectional view of a synthesized chest cavity.

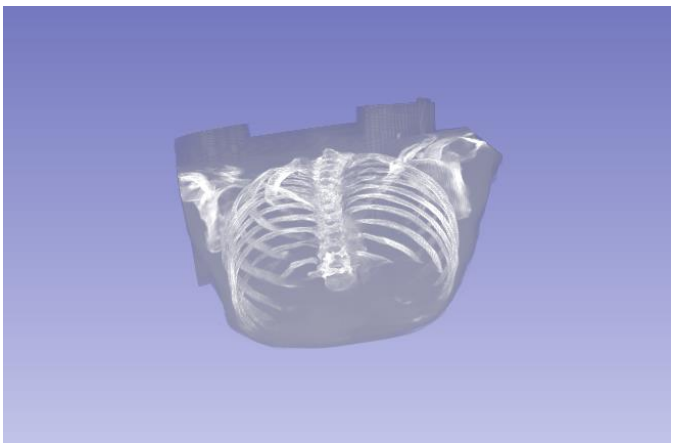

Figure 23: Rendering of the bone structure extracted from a synthetic 3D CT volume generated by our framework using OT-CFM, visualized using 3DSlicer.

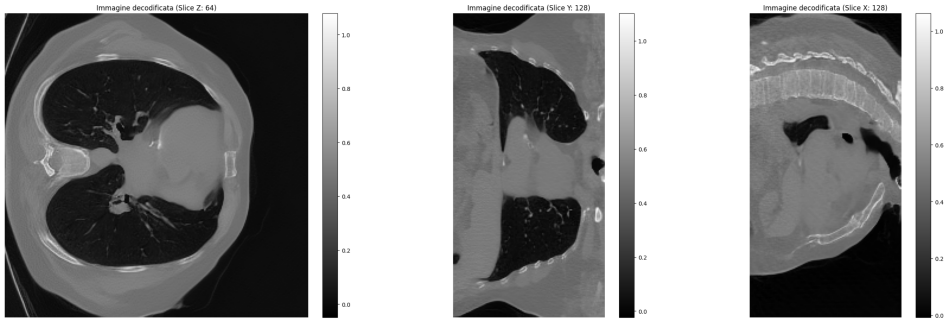

Figure 24: Cross-sectional slices of a successfully generated synthetic CT volume with OT-CFM.

Figures 25, 26, and 27 illustrate the evolution of the latent volumes from pure Gaussian noise to the final predicted data, representing the denoising process during the inference phase and Figures 28, 29 and 30 represent the variation in generation quality as the number of integration steps change for OT-CFM, I-CFM, and VP-CFM, respectively. Although specific successful selected samples demonstrate that all models can converge to anatomically plausible structures, robustness and consistency are not guaranteed for every generation.

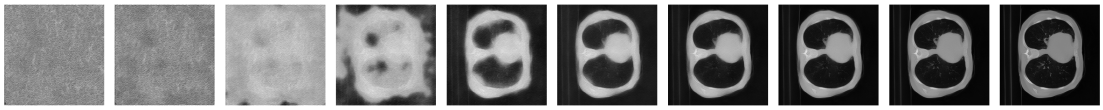

Figure 25: Denoising process during the inference phase with the denoising with OT-CFM

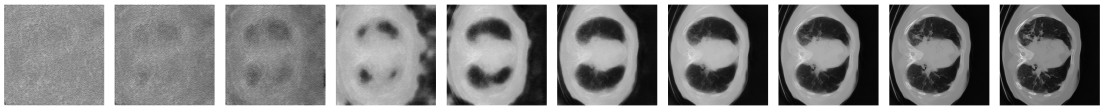

Figure 26: Denoising process during the inference phase with the denoising with I-CFM

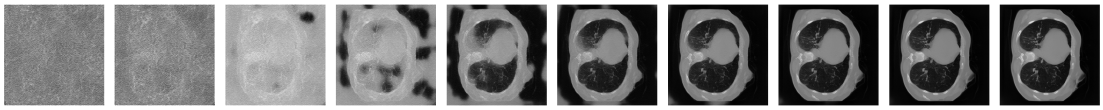

Figure 27: Denoising process during the inference phase with the denoising with VP-CFM

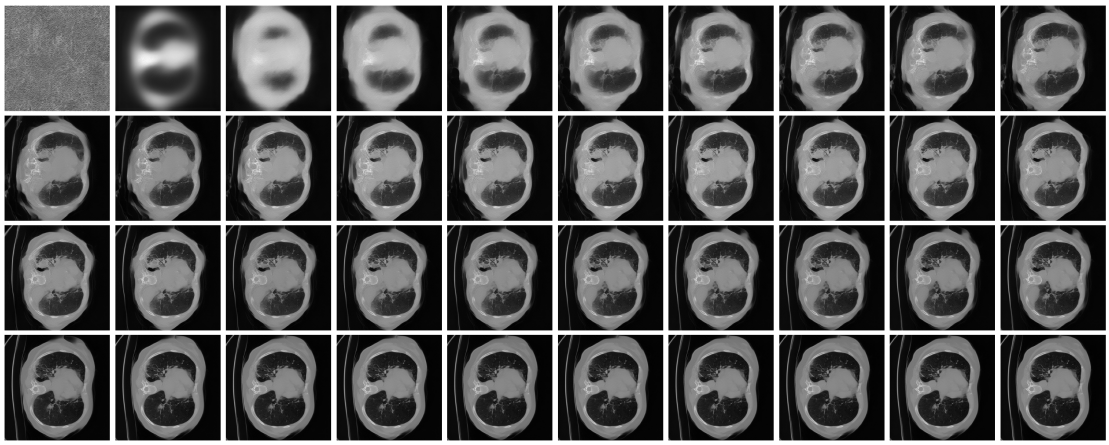

Figure 28: Quality of the generation through the number of integration step (using euler integrator) with OT-CFM. In the first three rows we have an increment of one step for each image, so from 1 to 30 steps. In the last row we have an increment of 20 steps for each image, so from 50 to 230.

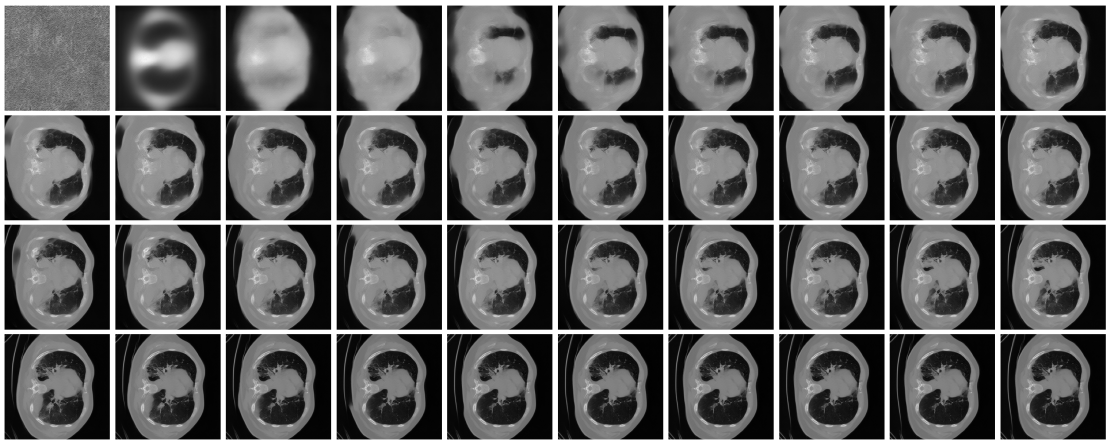

Figure 29: Quality of the generation through the number of integration step (using euler integrator) with I-CFM. In the first three rows we have an increment of one step for each image, so from 1 to 30 steps. In the last row we have an increment of 20 steps for each image, so from 50 to 230.

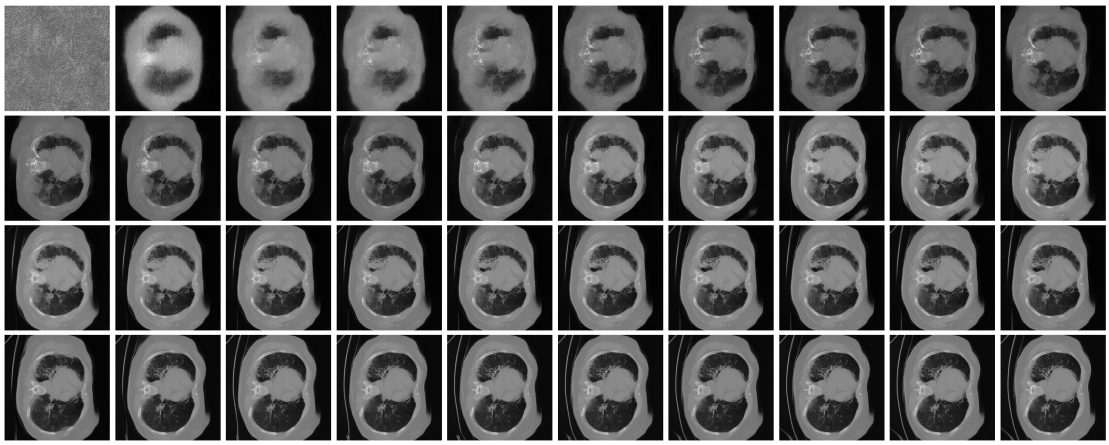

Figure 30: Quality of the generation through the number of integration step (using euler integrator) with VP-CFM. In the first three rows we have an increment of one step for each image, so from 1 to 30 steps. In the last row we have an increment of 20 steps for each image, so from 50 to 230.

Extensive generation of over 1000 samples revealed a critical disparity in the consistency of the generative process. While OT-CFM consistently produces sharp and highly detailed volumes across almost all random initializations, the generation quality of I-CFM and VP-CFM is highly dependent on the initial noise sample x_0.

For I-CFM and VP-CFM, the generation can more often degenerate in degraded results: while macroscopic characteristics and sharpness are maintained, the anatomical topology is frequently imprecise or totally violated. The most prominent artifacts include the misplacement of major structures, such as the spinal column collapsing into the lung cavities and the generation of morphologically inconsistent boundaries at the lung-air interfaces and distorted body contours (Fig. 31).

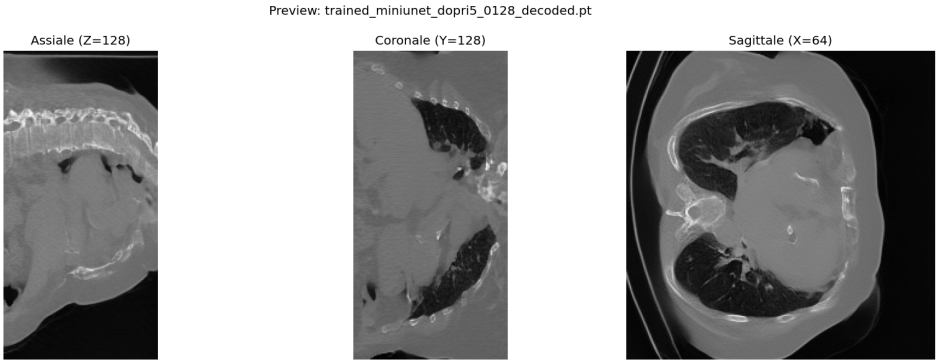

Figure 31: Example of a suboptimal generation obtained with I-CFM. Note the unnatural or undefined edges between the body and the air, blurred bones, and wavy textures of the anatomical components.

OT-CFM drastically reduces this spatial heterogeneity. By enforcing an Optimal Transport coupling within the minibatch, trajectories are disentangled globally. The variance of the regression target is uniformly minimized across the entire latent space, resulting in a vector field that is smoother, unidirectional, and constant over time. Consequently, the limited capacity of our network becomes sufficient to learn a much better physical dynamics everywhere, yielding consistently sharp and anatomically accurate volumes without dependency on the initial noise configuration.

5.2.1 The Loss-Fidelity Paradox: Regression Variance and Network Capacity

A critical empirical observation in our experiments is the apparent discrepancy between training metrics and inference performance. During the training phase, the Mean Squared Error (MSE) loss curves for OT-CFM, I-CFM, and VP-CFM exhibit similar convergence behaviors (Fig. 22), reaching comparable final values without significant divergence. Based solely on this optimization metric, one might theoretically expect all three models to achieve similar generative fidelity. However, as demonstrated by the qualitative degradation of I-CFM and VP-CFM, this theoretical assumption does not hold in practice.

The explanation of this phenomenon lies in analyzing the interaction between the target vector fields and the strict constraints of our neural network. The fundamental difference between the performance of models is the spatial variance of the regression target. In I-CFM and VP-CFM, the independent random coupling creates highly intersecting probability paths. At any given point in the latent space where multiple trajectories cross, the network receives conflicting target directions $u_t(x|z)$ during different training batches. To optimize computational efficiency, our 3D U-Net is intentionally constrained in capacity (with a maximum of 256 channels). Therefore, it lacks the structural bandwidth to memorize these highly entangled, high-variance target regions. Unable to regress the exact complex vector field, the network minimizes the MSE by predicting the spatial geometric average of the conflicting target paths. While this achieves a numerically low training loss, the resulting "averaged" vector does not align with an optimized physical trajectory. This mathematical compromise effectively averages the macroscopic positions of distinct anatomical features. If the network receives conflicting gradients directing a bone structure to two different valid spatial coordinates, it will generate a sharp bone structure in an invalid intermediate location. This "Mode Averaging" physically causes the topological anomalies and instability between our generated data, such as the spine collapsing into the lungs.

This diagnosis is supported by the scientific literature. Tong et al. [15] mathematically demonstrated that the introduction of Minibatch Optimal Transport "greatly reduces the variance of the regression target" by eliminating trajectory intersections. In OT-CFM, all trajectories passing through a local region share a consistent, linear direction. Furthermore, Tong et al. empirically proved that this variance reduction enables the use of smaller architectures without performance degradation, because a straight, low-variance target is significantly simpler to regress enabling computational optimization also from the point of view of the architecture of the neural network.

5.3 Quantitative Evaluation: Topological Fidelity and Mode Averaging

To validate our qualitative observations and provide an objective quantification of our synthetic datasets ($N = 1000$ samples per model), we evaluated them using three different metrics: the Fréchet Inception Distance (FID), the Maximum Mean Discrepancy (MMD), and the Voxel-wise Variance:

Fréchet Inception Distance (FID): The FID [25] evaluates the visual and semantic quality of the generated samples by comparing their feature representations against the real dataset. To extract the features of real and synthetic datasets, rather than relying on the standard ImageNet pre-trained networks, we use a ResNet50 architecture pre-trained on RadImageNet [26], a large-scale multi-modal radiological dataset. FID fits a continuous multivariate Gaussian to these extracted deep features and computes the Fréchet distance between the real and synthetic distributions. A lower FID score indicates higher topological and semantic fidelity.

Maximum Mean Discrepancy (MMD): The MMD [27] is a kernel-based statistical test that measures the distance between two probability distributions. It operates by mapping the real and synthetic feature distributions into an high-dimensional space and computing the distance between their mean embeddings. Unlike FID, which assumes a Gaussian distribution of the underlying features, MMD is a non-parametric metric that captures higher-order statistical discrepancies. In the context of generative models, a low MMD tell us that there is a good matching between the synthetic distribution and the real.

Voxel-wise Variance: To directly quantify the "Mode Averaging" hypothesis and measure inter-sample structural diversity, we computed the global Voxel-wise Variance. This metrics evaluates the statistical variance for each voxel across the entire generated dataset ($N = 1000$) and calculates the mean over the entire 3D volume. A low voxel-wise variance explicitly signals a structural collapse, indicating that the generative model is predicting a homogenized "average" anatomy rather than preserving true patient-to-patient topological diversity.

Model	**FID** (↓)	**MMD** (↓)	**Voxel-wise Variance** (↑)
OT-CFM	135	0.006	0.4642 ± 0.0015
I-CFM	5435	0.099	0.4412 ± 0.0014
VP-CFM	6265	0.103	0.4655 ± 0.0016

Table 4: Quantitative Evaluation of 3D Medical Volume Generation. FID and MMD evaluate semantic and topological fidelity compared to the real data distribution. The Voxel-wise Variance (reported with Bootstrap Standard Error, $N_{boot} = 100$) quantifies the inter-sample diversity.

As shown in Table 4, while OT-CFM achieves a good FID of 135 and an MMD of 0.006, both I-CFM and VP-CFM exhibit a greater degradation, with FID scores that are orders

of magnitude greater. This divergence mathematically confirms that the anomalies observed in I-CFM and VP-CFM (e.g., misplaced spinal columns and distorted lung boundaries) are frequent and systemic, not rare cases as in OT-CFM.

To investigate the diversity of the generated images, we computed the Voxel-wise Variance across the generated datasets. This metric reveals that I-CFM suffers from a significant drop in structural diversity, collapsing to a variance of 0.4412 ± 0.0014 compared to the 0.4642 ± 0.0015 baseline of OT-CFM. This significant reduction can be interpreted as Mode Averaging. Confused by the highly intersecting, conflicting target paths generated by independent couplings, our capacity-constrained 3D U-Net minimized the global MSE loss by predicting the spatial mean of the target anatomies. Consequently, the network avoided generating extreme, patient-specific topological variations, retreating instead towards a standardized, "average" latent structure. Because the geometric average of two valid anatomical positions often falls into an invalid intermediate space (e.g., inside a lung cavity), this mode averaging directly causes the severe topological penalties measured by the FID.

Interestingly, while VP-CFM maintains a higher variance (0.4655 ± 0.0016) due to its inherent variance-preserving formulation, its poor FID score confirms that this metric reflects scattered structural artifacts and topological instability rather than coherent anatomical diversity, a failure which is however due to the highly intersecting trajectories of its independent coupling.

5.4 Future Works

The results presented in this thesis open several possible avenues for future research across different disciplines:

- **Clinical Perspective - Conditioned Generation:** While our work successfully optimized the unconditional generative core, a natural progression is the integration and validation of the spatial conditioning framework originally implemented by the MAISI team (ControlNet). Adapting this conditioning mechanism to our optimized OT-CFM model would enable the targeted synthesis of specific pathological conditions (e.g., generating lung nodules or lesions at precise anatomical coordinates), bridging the gap between theoretical generation and practical downstream clinical applications.

- **Computational Perspective - From 3D to 2D Sequential Generation:** To further democratize generative AI and drastically reduce hardware limitations, future work could explore transitioning from a native 3D architecture to a 2D slice-wise generation paradigm. The primary mathematical and architectural challenge in this context would be enforcing strict spatial and topological coherence along the longitudinal axis (Z-axis). This could potentially be addressed by introducing cross-slice attention mechanisms or sequential conditioning within the continuous Flow Matching dynamics.

- **Theoretical Physics Perspective - Vector Field Dynamics in Capacity-Constrained Regimes:** The empirical observation of the "Mode Averaging" phenomenon highlights a fundamental issue in continuous generative modeling: the discrepancy between the theoretical target vector field and the empirical vector field learned by an under-parameterized function approximator. A rigorous dynamical systems analysis of the empirical vector field could reveal the general laws governing this topological degradation, providing deeper insights into how architectural bottlenecks physically smooth and distort complex theoretical trajectories. Further research in this area could drive progress from multiple perspectives, helping to develop and refine the generative framework theoretically and computationally to improve both performance and efficiency, avoiding the brute-force approach of simply scaling up model size.

Conclusions

This thesis addressed the significant computational and architectural challenges associated with the generation of high-resolution 3D medical volumes. By leveraging the MAISI framework, we transitioned the generative process from the high-dimensional pixel space into a compressed latent space, subsequently exploring the mathematical and practical implications of different continuous-time generative dynamics.

Our investigation revealed that while modern Flow Matching techniques offer a generalized and efficient alternative to standard diffusion models, their performance is fundamentally tied to the geometric properties of their target probability paths, especially under strict network capacity constraints. Through extensive empirical evaluation on the CT-RATE dataset, we uncovered a critical failure mode inherent to random independent couplings (I-CFM) and diffusion-inspired schedules (VP-CFM). When forced to regress the highly intersecting and conflicting trajectories generated by these methods, our lightweight 3D U-Net minimized the empirical loss by defaulting to the spatial, geometric average of the conflicting anatomical structures.

This phenomenon, which we formally diagnosed as "Mode Averaging," physically manifested as topological anomalies, such as collapsed spinal columns and distorted organ boundaries. This visual degradation was quantitatively confirmed by a severe drop in Voxel-wise Variance and high Fréchet Inception Distance (FID) and Maximum Mean Discrepancy (MMD) scores, proving that a low training loss does not guarantee structural fidelity in under-parameterized regimes.

To resolve this capacity bottleneck without resorting to the computationally expensive approach of scaling up the neural architecture, we successfully implemented Minibatch Optimal Transport Conditional Flow Matching (OT-CFM). By optimizing the coupling between source noise and target data distributions, OT-CFM effectively disentangles the probability paths globally, enforcing straight, non-crossing trajectories. Our results definitively proved that this spatial variance reduction transforms the regression task into a tractable problem for the constrained network. Consequently, OT-CFM consistently generated sharp, topologically coherent, and diverse 3D synthetic CT volumes, outperforming the baselines across all semantic and structural metrics.

Ultimately, this work demonstrates that Optimal Transport is not merely an optional convergence accelerator, but a strict mathematical and architectural prerequisite for deploying continuous generative models in resource-constrained 3D applications. By aligning the theoretical vector field with the physical limits of the network, we provided a highly efficient, scalable framework capable of democratizing synthetic medical data generation for broader clinical and research applications.

Appendix

A Marginal Vector Field Aggregation

Let $p_t(x|z)$ be a conditional probability path generated by a vector field $u_t(x|z)$, and let $q(z)$ be a distribution over the conditioning variable z. The marginal probability path $p_t(x) = \int p_t(x|z)q(z)dz$ is generated by the marginal vector field $u_t(x)$, defined as:

$$u_t(x) = \mathbb{E}_{q(z|x)}[u_t(x|z)] = \frac{1}{p_t(x)} \int u_t(x|z)p_t(x|z)q(z)dz \tag{36}$$

Proof. The continuity equation for the marginal path is:

$$\frac{\partial p_t(x)}{\partial t} + \nabla \cdot (p_t(x)u_t(x)) = 0 \tag{37}$$

By linearity of the continuity equation, we substitute the definition of the marginal density $p_t(x) = \int p_t(x|z)q(z)dz$:

$$\frac{\partial}{\partial t} \int p_t(x|z)q(z)dz + \nabla \cdot (p_t(x)u_t(x)) = 0 \tag{38}$$

Since $p_t(x|z)$ satisfies the continuity equation individually with its generating field $u_t(x|z)$, we have $\frac{\partial p_t(x|z)}{\partial t} = -\nabla \cdot (p_t(x|z)u_t(x|z))$. Substituting this back into the integral:

$$\int -\nabla \cdot (p_t(x|z)u_t(x|z))q(z)dz + \nabla \cdot (p_t(x)u_t(x)) = 0 \tag{39}$$

Exchanging the order of integration and divergence operators:

$$\nabla \cdot \left(p_t(x)u_t(x) - \int u_t(x|z)p_t(x|z)q(z)dz \right) = 0 \tag{40}$$

For this equation to hold for any distribution $p_t(x)$, the term inside the divergence must be zero (assuming a curl-free field for uniqueness), which yields:

$$p_t(x)u_t(x) = \int u_t(x|z)p_t(x|z)q(z)dz \tag{41}$$

Dividing by $p_t(x)$ completes the proof [14].

□

B Equivalence of Objectives Proof

Assuming that $p_t(x) > 0$ for all $x \in \mathbb{R}^d$ and $t \in [0, 1]$, then, up to a constant independent of θ, $\mathcal{L}_{CFM}$ and $\mathcal{L}_{FM}$ are equal. Hence, $\nabla_\theta \mathcal{L}_{FM}(\theta) = \nabla_\theta \mathcal{L}_{CFM}(\theta)$.

Proof. To ensure the existence of all integrals and to allow the changing of integration order (by Fubini's Theorem) in the following we assume that $q(z)$ and $p_t(x|z)$ are decreasing to zero at a sufficient speed as $||x|| \to \infty$, and that $u_t, v_t, \nabla_\theta v_t$ are bounded.

First, using the standard bilinearity of the 2-norm we have that:

$$||v_t(x) - u_t(x)||^2 = ||v_t(x)||^2 - 2\langle v_t(x), u_t(x)\rangle + ||u_t(x)||^2 \tag{42}$$

$$||v_t(x) - u_t(x|z)||^2 = ||v_t(x)||^2 - 2\langle v_t(x), u_t(x|z)\rangle + ||u_t(x|z)||^2 \tag{43}$$

Integrating the first equation with respect to $p_t(x)$ yields $\mathcal{L}_{FM}$ (up to the expectation over t):

$$\mathbb{E}_{p_t(x)}[||v_t(x) - u_t(x)||^2] = \mathbb{E}_{p_t(x)}[||v_t(x)||^2] - 2\mathbb{E}_{p_t(x)}[\langle v_t(x), u_t(x)\rangle] + C_1 \tag{44}$$

where $C_1 = \mathbb{E}_{p_t(x)}[||u_t(x)||^2]$ is independent of θ.

Integrating the second equation with respect to $q(z)p_t(x|z)$ yields $\mathcal{L}_{CFM}$:

$$\begin{aligned}\mathbb{E}_{q(z)p_t(x|z)}[||v_t(x) - u_t(x|z)||^2] = \mathbb{E}_{q(z)p_t(x|z)}[||v_t(x)||^2] \\ - 2\mathbb{E}_{q(z)p_t(x|z)}[\langle v_t(x), u_t(x|z)\rangle] + C_2\end{aligned} \tag{45}$$

where $C_2 = \mathbb{E}_{q(z)p_t(x|z)}[||u_t(x|z)||^2]$ is independent of θ.

Now we show that the terms dependent on θ in both objectives are identical. For the quadratic term $||v_t(x)||^2$:

$$\mathbb{E}_{q(z)p_t(x|z)}[||v_t(x)||^2] = \int q(z) \int p_t(x|z)||v_t(x)||^2 \, dx \, dz \tag{46}$$

Using $p_t(x) = \int p_t(x|z)q(z)dz$, we can swap the order of integration:

$$= \int ||v_t(x)||^2 \left(\int p_t(x|z)q(z) \, dz \right) dx = \int ||v_t(x)||^2 p_t(x) \, dx = \mathbb{E}_{p_t(x)}[||v_t(x)||^2] \tag{47}$$

For the cross term $\langle v_t(x), u_t(x|z)\rangle$:

$$\mathbb{E}_{q(z)p_t(x|z)}[\langle v_t(x), u_t(x|z)\rangle] = \int v_t(x)^T \left(\int u_t(x|z)p_t(x|z)q(z) \, dz \right) dx \tag{48}$$

Using the definition of the marginal vector field $u_t(x) = \frac{1}{p_t(x)} \int u_t(x|z)p_t(x|z)q(z)dz$ (Theorem 1), we substitute the term in the parenthesis:

$$= \int v_t(x)^T (u_t(x)p_t(x)) \, dx = \mathbb{E}_{p_t(x)}[\langle v_t(x), u_t(x)\rangle] \tag{49}$$

Since both the quadratic and cross terms match, the gradients $\nabla_\theta \mathcal{L}_{FM}$ and $\nabla_\theta \mathcal{L}_{CFM}$ are [14].

□

C Derivation of the Probability Flow ODE from CFM

Consider the Gaussian probability path defined by the Variance Preserving SDE noise schedule β_t. The conditional distribution $p_t(x|x_1) = \mathcal{N}(x; \mu_t(x_1), \sigma_t^2 I)$ is characterized by the following mean and standard deviation:

$$\mu_t(x_1) = \alpha_t x_1 \tag{50}$$

$$\sigma_t = \sqrt{1 - \alpha_t^2} \tag{51}$$

where α_t is defined such that its time evolution satisfies the VP-SDE drift:

$$\dot{\alpha}_t = -\frac{1}{2}\beta_t \alpha_t \tag{52}$$

Consequently, the time derivative of the standard deviation follows:

$$\dot{\sigma}_t = \frac{d}{dt}\sqrt{1 - \alpha_t^2} = \frac{-2\alpha_t \dot{\alpha}_t}{2\sqrt{1 - \alpha_t^2}} = \frac{-\alpha_t(-\frac{1}{2}\beta_t \alpha_t)}{\sigma_t} = \frac{1}{2}\beta_t \frac{\alpha_t^2}{\sigma_t} \tag{53}$$

Recall that for a general Gaussian conditional path $p_t(x|x_1)$, the unique vector field generating it is given by:

$$u_t(x|x_1) = \frac{\dot{\sigma}_t}{\sigma_t}(x - \mu_t(x_1)) + \dot{\mu}_t(x_1) \tag{54}$$

Substituting the VP-SDE specific derivatives into Equation 54:

$$u_t(x|x_1) = \underbrace{\left(\frac{1}{2}\beta_t \frac{\alpha_t^2}{\sigma_t^2}\right)}_{\dot{\sigma}_t/\sigma_t}(x - \alpha_t x_1) + \underbrace{\left(-\frac{1}{2}\beta_t \alpha_t x_1\right)}_{\dot{\mu}_t} \tag{55}$$

To simplify, we factor out $-\frac{1}{2}\beta_t$:

$$u_t(x|x_1) = -\frac{1}{2}\beta_t \left[-\frac{\alpha_t^2}{\sigma_t^2}(x - \alpha_t x_1) + \alpha_t x_1\right] \tag{56}$$

We now relate the displacement term to the score function of the conditional distribution, $\nabla \log p_t(x|x_1)$. For a Gaussian, the score is given by:

$$\nabla \log p_t(x|x_1) = -\frac{x - \alpha_t x_1}{\sigma_t^2} \tag{57}$$

Inverting this relationship allows us to express the clean data x_1 in terms of the noisy data x and the score (Tweedie's formula):

$$\alpha_t x_1 = x + \sigma_t^2 \nabla \log p_t(x|x_1) \tag{58}$$

Substituting these identities into the bracketed term of Equation 56:

$$[\dots] = -\alpha_t^2 \underbrace{\left(\frac{x - \alpha_t x_1}{\sigma_t^2}\right)}_{-\nabla \log p_t(x|x_1)} + \underbrace{(\alpha_t x_1)}_{x + \sigma_t^2 \nabla \log p_t(x|x_1)} \tag{59}$$

$$= \alpha_t^2 \nabla \log p_t(x|x_1) + x + \sigma_t^2 \nabla \log p_t(x|x_1) \tag{60}$$

$$= x + (\alpha_t^2 + \sigma_t^2) \nabla \log p_t(x|x_1) \tag{61}$$

Since $\alpha_t^2 + \sigma_t^2 = 1$ by definition, the conditional vector field simplifies to:

$$u_t(x|x_1) = -\frac{1}{2}\beta_t \left[x + \nabla \log p_t(x|x_1)\right] \tag{62}$$

Finally, to find the marginal vector field $v_t(x)$ that generates the marginal path $p_t(x)$, we take the expectation over the posterior $p(x_1|x)$ (recalling the aggregation property derived in Appendix A):

$$v_t(x) = \mathbb{E}_{x_1 \sim p(x_1|x)}[u_t(x|x_1)] \tag{63}$$

$$= -\frac{1}{2}\beta_t \left[x + \mathbb{E}_{x_1 \sim p(x_1|x)}[\nabla \log p_t(x|x_1)]\right] \tag{64}$$

Using the identity $\nabla \log p_t(x) = \mathbb{E}_{x_1 \sim p(x_1|x)}[\nabla \log p_t(x|x_1)]$, we arrive at the final form:

$$v_t(x) = -\frac{1}{2}\beta_t \left[x + \nabla \log p_t(x)\right] \tag{65}$$

This equation is exactly the Probability Flow ODE associated with the VP-SDE, as presented in Song et al. [28].

Use of generative AI

Generative AI was used during this thesis project for didactical support, brainstorming, text revision, and coding assistance. All AI-generated content was subjected to severe supervision and critical review by the author prior to inclusion.

References

[1] Z. Li, Z. Zeng, X. Lin, F. Fang, Y. Qu, Z. Xu, Z. Liu, X. Ning, T. Wei, G. Liu, *et al.*, "Flow matching meets biology and life science: a survey," *npj Artificial Intelligence*, vol. 2, no. 1, p. 17, 2026.

[2] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *CoRR*, vol. abs/2003.05991, 2020.

[3] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[4] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "Ffjord: Free-form continuous dynamics for scalable reversible generative models," *arXiv preprint arXiv:1810.01367*, 2018.

[5] T. Fjelde, E. Mathieu, and V. Dutordoir, "An introduction to flow matching," January 2024.

[6] P. Guo, C. Zhao, D. Yang, Z. Xu, V. Nath, Y. Tang, B. Simon, M. Belue, S. Harmon, B. Turkbey, *et al.*, "Maisi: Medical ai for synthetic imaging," in *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 4430–4441, IEEE, 2025.

[7] "Rectified flow." `https://github.com/gnobitab/RectifiedFlow`. 2026-03-26.

[8] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," *arXiv preprint arXiv:2209.03003*, 2022.

[9] G. N. Hounsfield, "Computerized transverse axial scanning (tomography): Part 1. description of system," *The British journal of radiology*, vol. 46, no. 552, pp. 1016–1022, 1973.

[10] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

[11] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[12] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[13] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[14] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.02747*, 2022.

[15] A. Tong, K. Fatras, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, G. Wolf, and Y. Bengio, "Improving and generalizing flow-based generative models with minibatch optimal transport," *arXiv preprint arXiv:2302.00482*, 2023.

[16] R. J. McCann, "A convexity principle for interacting gases," *Advances in mathematics*, vol. 128, no. 1, pp. 153–179, 1997.

[17] C. Villani *et al.*, *Optimal transport: old and new*, vol. 338. Springer, 2008.

[18] J.-D. Benamou and Y. Brenier, "A computational fluid mechanics solution to the monge-kantorovich mass transfer problem," *Numerische Mathematik*, vol. 84, no. 3, pp. 375–393, 2000.

[19] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[20] C. Zhao, P. Guo, D. Yang, Y. Tang, Y. He, B. Simon, M. Belue, S. Harmon, B. Turkbey, and D. Xu, "Maisi-v2: Accelerated 3d high-resolution medical image synthesis with rectified flow and region-specific contrastive loss," *arXiv preprint arXiv:2508.05772*, 2025.

[21] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3836–3847, 2023.

[22] M. J. Cardoso, W. Li, R. Brown, N. Ma, E. Kerfoot, Y. Wang, B. Murrey, A. Myronenko, C. Zhao, D. Yang, *et al.*, "Monai: An open-source framework for deep learning in healthcare," *arXiv preprint arXiv:2211.02701*, 2022.

[23] MONAI, "Project monai." `https://github.com/Project-MONAI/MONAI`. 2026-03-26.

[24] I. E. Hamamci, S. Er, F. Almas, A. G. Simsek, S. N. Esirgun, I. Dogan, M. F. Dasdelen, B. Wittmann, E. Simsar, M. Simsar, E. B. Erdemir, A. Alanbay, A. Sekuboyina, B. Lafci, M. K. Özdemir, and B. H. Menze, "A foundation model utilizing chest ct volumes and radiology reports for supervised-level zero-shot detection of abnormalities," *CoRR*, vol. abs/2403.17834, 2024.

[25] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[26] X. Mei, Z. Liu, P. M. Robson, B. Marinelli, M. Huang, A. Doshi, A. Jacobi, C. Cao, K. E. Link, T. Yang, *et al.*, "Radimagenet: an open radiologic deep learning research dataset for effective transfer learning," *Radiology: Artificial Intelligence*, vol. 4, no. 5, p. e210315, 2022.

[27] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *The journal of machine learning research*, vol. 13, no. 1, pp. 723–773, 2012.

[28] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.