TECHNISCHE UNIVERSITÄT DRESDEN

HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

Faculty of Computer Science

Helmholtz-Zentrum Dresden-Rossendorf - Institute of Radiation Physics
Department of Laser Particle Acceleration

Master Thesis

# Adapting gated normalizing flows-based architecture for point-cloud generation to surrogate modeling of particle dynamics

Gopal Damodar Kulkarni

Born on: 22nd October 1995 in Hubballi, Karnataka
Matriculation number: 4991300

10th July 2024

First referee
Prof. Dr. Thomas Cowan

Second referee
Prof. Dr. Ulrich Schramm

Supervisor
Dr. Jeffrey Kelling

TECHNISCHE UNIVERSITÄT DRESDEN

HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

**Faculty of Computer Science** Helmholtz-Zentrum Dresden-Rossendorf - Institute of Radiation Physics

Department of Laser Particle Acceleration

## Task for the preparation of a Master Thesis

| | |
|---|---|
| Course: | Computational Modeling and Simulation |
| Name: | Gopal Damodar Kulkarni |
| Matriculation number: | 4991300 |
| Matriculation year: | 2020 |
| Title: | Adapting gated normalizing flows-based architecture for point-cloud generation to surrogate modeling of particle dynamics |

### Objectives of work

This work systematically investigates the previously-proposed mixture of normalizing flows approach, which promises to enable normalizing flow (NF)-based generative models to better learn multi-modal posterior distributions with conditionality to enable surrogate modeling. A systematic study of appropriately designed test distributions to provide insights into specific advantages and limitations of the proposed approach and an analysis of how these may be connected to choices made for interchangeable blocks during implementation is carried out. Further, a study of options to balance training of encoder and decoder stages towards reducing overall training time and memory consumption, including an analysis of possible performance-quality trade-offs is carried out. Finally, the model with the best configuration in the above studies is applied to available simulation data of particle propagation.

### Focus of work

- Study of the theoretical background and the related literature works.
- Understanding the behavior of the model with test distributions
- Formulation of training protocols and model reduction techniques to reduce the computational costs
- Application of the model to simulation data of particle propagation
- Analysis and evaluation of the results

| | |
|---|---|
| First referee: | Prof. Dr. Thomas Cowan |
| Second referee: | Prof. Dr. Ulrich Schramm |
| Supervisor: | Dr. Jeffrey Kelling |
| Issued on: | 7th February 2024 |
| Due date for submission: | 10th July 2024 |

Prof. Dr. Thomas Cowan
Supervising professor

# Statement of authorship

I hereby certify that I have authored this document entitled *Adapting gated normalizing flows-based architecture for point-cloud generation to surrogate modeling of particle dynamics* independently and without undue assistance from third parties. No other than the resources and references indicated in this document have been used. I have marked both literal and accordingly adopted quotations as such. There were no additional persons involved in the intellectual preparation of the present document. I am aware that violations of this declaration may lead to subsequent withdrawal of the academic degree.

Dresden, 10th July 2024

Gopal Damodar Kulkarni

Digitally signed by Gopal Damodar Kulkarni
Date: 2024.07.10 10:48:33 +02'00'

Gopal Damodar Kulkarni

TECHNISCHE UNIVERSITÄT DRESDEN

HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

**Faculty of Computer Science** Helmholtz-Zentrum Dresden-Rossendorf - Institute of Radiation Physics

Department of Laser Particle Acceleration

## Abstract

Plasma acceleration processes have garnered extensive research interest in recent years due to the versatile applications of high-energy X-rays, including medical diagnostics and treatment, semiconductor manufacturing, and the hardening of material surfaces. However, simulating these processes demands a complex workflow and considerable computational time, rendering in-situ analysis impractical. To mitigate these challenges, simulation-based inference and data-driven methods for one-step inversion can be utilized to recover the matching phase-space representation and elucidate the underlying physics.

This work presents the adaptation of a surrogate model specifically for plasma physics processes, such as beam transport in a Free Electron Laser (FEL). A previously proposed model, which uses a mixture of normalizing flows to learn multi-modal distributions, is systematically investigated. Conditionality as a means of surrogate modeling in plasma physics is investigated and trained on simulation results of plasma physics processes.

Initial experiments involved using various custom test distributions to derive key insights into the model's operation. Conditionality was incorporated into the model by utilizing the single-view reconstruction provision appropriately, allowing it to handle a wider range of input conditions. This conditioning has been introduced in two forms, scalar conditioning, and one-hot vector conditioning, and the effectiveness of each method is studied. This redefined architecture was rigorously tested on various test distributions with differing characteristics to validate its performance. The reconstruction quality was then compared to point clouds generated by single normalizing flow-based models with similar network sizes. Additionally, a significant reduction in trainable parameters was achieved, making the model more computationally feasible. Custom training protocols were introduced to further reduce overall training time.

The enhanced architecture was subsequently applied to simulation data from a free electron laser. A comprehensive analysis of the results demonstrated the surrogate model's capability to accurately capture the complex dynamics of plasma acceleration processes. This work highlights the potential of advanced surrogate models in reducing computational demands and providing deeper insights into plasma physics, paving the way for more efficient and practical applications in high-energy X-ray technologies.

# Contents

Abstract . . . VII

1 Introduction . . . 1
1.1 Motivation: Plasma physics and applications . . . 1
1.2 Simulations in plasma physics . . . 2
1.2.1 Research Goals: Surrogate modeling . . . 2
1.3 Report Overview . . . 4

2 Literature Review: Generative models for point clouds . . . 5

3 Theoretical Background . . . 7
3.1 Threshold Logic Units . . . 7
3.2 Multilayered Perceptrons (MLP) . . . 8
3.3 Autoencoders . . . 9
3.4 Variational Autoencoders . . . 9
3.5 Normalizing Flows . . . 10
3.6 Mixture of Normalizing flows . . . 12
3.6.1 Loss Function . . . 13
3.7 Sinkhorn Distance . . . 14
3.7.1 Optimal Transport Problem . . . 14
3.7.2 Entropy Regularization . . . 15
3.7.3 Sinkhorn Algorithm . . . 15

4 Experimentation on Mixture of NFs . . . 17
4.1 Model behavior on simple geometries as input data . . . 17
4.2 Architectural Modification for conditionality . . . 20
4.3 Comparative Analysis: Mixture of NFs versus Single NF-based Model . . . 22
4.3.1 Higher Dimensional Input . . . 25
4.3.2 Variation in Density . . . 29
4.4 Model behavior with increased number of conditions . . . 30
4.4.1 Analysis of the learned latent distributions . . . 30

5 Model Reduction and Training Protocols . . . 33
5.1 Model Reduction . . . 33

5.2 Training Protocols . . . . . 36

6 Application of Mixture of NFs to Plasma Physics Process . . . . . 39
6.1 Free-Electron Laser . . . . . 39
6.2 Generated data: Scalar Conditioning . . . . . 40
6.3 Generated data: One-hot Vector Conditioning . . . . . 44

7 Conclusion . . . . . 47
7.1 Discussion . . . . . 47
7.2 Summary . . . . . 48
7.3 Outlook . . . . . 49

# 1 Introduction

## 1.1 Motivation: Plasma physics and applications

Plasma, recognized as the fourth state of matter, emerges at high energy levels. Typically, when atoms in a solid gain sufficient energy, they transition to the liquid phase, and with further energy input, they move to the gas phase. At extremely high temperatures, plasma is formed. In this state, some or all of the electrons in an atom acquire enough energy to overcome the potential barrier of the nucleus, resulting in an ionized gas.

However, not all ionized gases qualify as plasma, since a small degree of ionization is present in any gas. Therefore, a more precise and functional definition is as follows:

A plasma is a quasi-neutral gas composed of charged and neutral particles demonstrating collective behavior. "Quasineutral" implies that the plasma is generally electrically neutral with free charges, essential for maintaining significant electromagnetic interactions. "Collective behavior" refers to motions that depend not only on local conditions but also on the state of the plasma in distant regions.

Due to its high energy and versatility, plasma has a broad range of technological applications. Its unique properties, such as high electrical conductivity, responsiveness to electric and magnetic fields, and collective behavior, make it relevant in science and technology. Some notable applications include:

1. Fusion Energy: Nuclear fusion involves the merging of two atomic nuclei to form a heavier nucleus, releasing a significant amount of energy. Fusion power generation necessitates maintaining a 30 keV plasma plasma within a magnetic field for at least one second.
2. Surface Modifications: Plasma surface treatment systems enhance product quality by improving hardness, corrosion resistance, bio-compatibility, and altering optical and electronic properties. A major application is in the fabrication of integrated circuits (ICs), where plasma processes account for about one-third of semiconductor production steps.

3. **Laser-driven Particle Accelerators:** These accelerators offer promising applications in fundamental research in particle physics, medical imaging and therapy, materials science, and high-energy-density physics. These lasers are used to ionize a target material and appropriate mechanisms are used to cause an accelerated bunch of charged particles to be ejected. Their compact size and capability for achieving high energies make them ideal for applications where traditional accelerators are impractical or too expensive.

## 1.2 Simulations in plasma physics

Simulations are mathematical models or representations of real-world systems or phenomena, designed to mimic their behavior under specific conditions. This allows researchers to study, analyze, and understand complex processes in a controlled environment. In plasma physics, these mathematical models fall into two categories: particle models and fluid models.

Particle models, such as the Particle-in-Cell (PIC) method, attempt to replicate nature by tracking the motion of a large number of charged particles within their self-consistent electric and magnetic fields. These models capture plasma behavior on the finest spatial scales and the most rapid time scales.

In contrast, fluid models use a set of fluid equations to describe plasma behavior, typically on larger spatial and temporal scales. An example of a fluid model is the Magnetohydrodynamics (MHD) model.

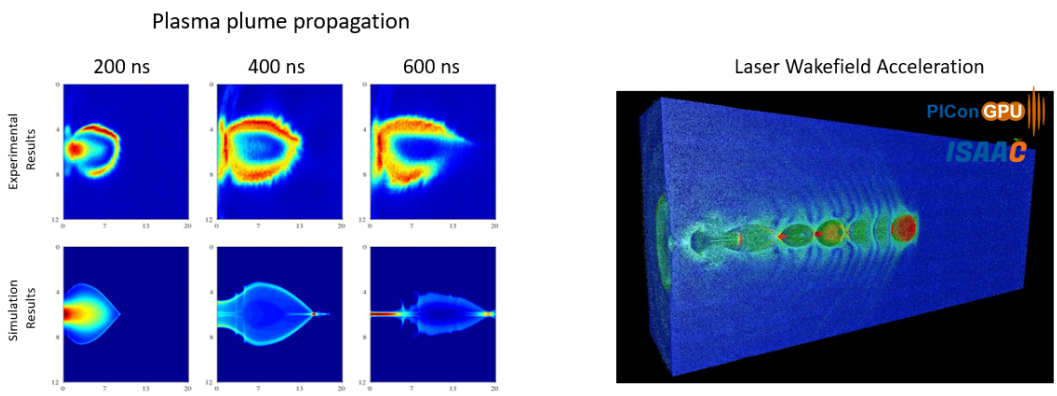


Figure 1.1: Simulations of plasma physics processes are shown in this figure. The MHD simulation of a plasma plume propagation transverse to an applied magnetic field is shown on the left [Pat+21]. PIC (Particle in-cell) Simulation of the laser wakefield acceleration process is shown on the right [Wil22]

### 1.2.1 Research Goals: Surrogate modeling

While these physics-based simulations of processes in plasma physics can improve our understanding and prediction capacities, these first-principle simulations can be computationally expensive. For example, simulation time for physical instabilities using gyrokinetic particle-in-cell codes is often on the order of hours on modern GPUs [Mad+11].

On the other hand, statistical methods including machine learning models have achieved promising results [Don+21]. Accelerated simulations by surrogate modeling techniques are carried out in two main directions [Sti+20]: Unsupervised learning methods require full simulation data to train some neural network architecture, e.g. generative adversarial networks [Goo+14] or autoencoders [KW22], to reproduce numerical simulations with the benefits of interpolation between two similar configurations. Another way of surrogate modeling is through self-supervised methods by either embedding neural networks in the numerical procedures for solving the governing equations or incorporating information about the governing equations in the loss functions of the neural networks which are more popularly known as PINNs (Physics Informed Neural Networks) [Sti+20]. The primary advantage of such a data-driven surrogate model over the first-principle physics-based simulation is that it is better suited for in-situ analysis of the experimental results.

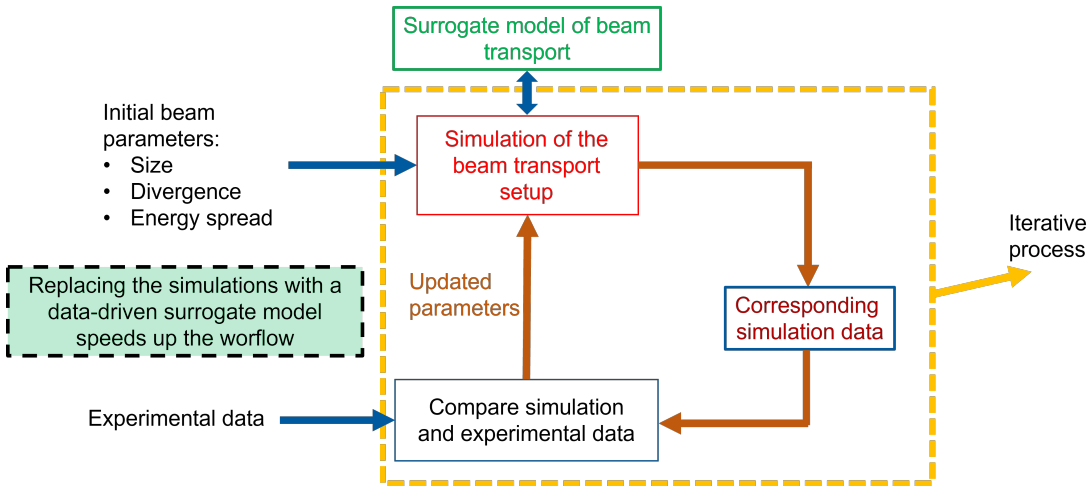


Figure 1.2: A workflow leveraging machine learning techniques to determine the most effective FEL simulation parameters, aiming to reconcile disparities between experimental and simulated outcomes

Fig. [1.2] describes a machine-learning-based workflow that employs a surrogate model instead of simulations to infer the right set of parameters in the case of a Free Electron Laser (FEL) simulation. The simulations initialized with the beam parameters are compared with the experimental results. To bridge the gap between the physical and simulation data, the appropriate set of simulation parameters needs to be determined. This is done by iteratively improving the simulation parameters. Instead of performing entire simulations on the new sets of parameters, a data-driven surrogate model is used to infer the simulation results for the new parameters which can significantly improve the workflow.
In this work, the first approach to formulate a surrogate model of plasma physics simulations by training a generative model on simulation data of plasma processes is taken. More specifically, a generative model that learns from and produces data in the form of point clouds is sought to be developed.

## 1.3 Report Overview

This project aims to achieve the established research goals by utilizing the architecture of the Mixture of normalizing flows as the foundation for the surrogate model. The results of this endeavor are documented in this report, which is organized into chapters and sections detailed below.

Chapter 3 introduces the theoretical foundations of several fundamental machine learning models and algorithms, which constitute the generative model extensively utilized in this work. These algorithms serve as the fundamental building blocks of the model employed in this project. This includes the algorithm of a threshold logic unit, architectures of Variational Autoencoders, normalizing flows, and the Mixture of normalizing flows.

In the subsequent Chapter 2, contemporary methodologies are examined, focusing on various models utilized for learning and generating distributions of point clouds. Notable architectures, including Generative Adversarial Networks (GAN) and Variational Autoencoders (VAE), have been repurposed for point cloud applications, each presenting its unique set of challenges. Additionally, this chapter provides a concise overview of more advanced frameworks that have surfaced in recent years.

Chapter 4 delves into a series of experiments designed to explore the behavior of the Mixture of normalizing flows architecture. Initially, experiments focus on studying the decoder distributions using test distributions. Subsequent experiments involve architectural modifications aimed at introducing the feature of conditionality to the framework. These modifications are then thoroughly investigated through further experiments, with subsequent analysis aimed at elucidating the observed characteristics of the model.

The importance of feasibility in terms of computational costs cannot be overstated, particularly since the primary objective of a data-driven surrogate model is to minimize these costs. Chapter 5 is dedicated to achieving this goal through rigorous model reduction techniques and the implementation of custom training protocols. These efforts culminate in a substantial reduction of trainable parameters, ultimately leading to improvements in both training time and computational efficiency. By prioritizing these aspects, the chapter aims to enhance the practicality and scalability of the proposed surrogate model.

In Chapter 6, the developed surrogate model is applied to reconstruct point clouds corresponding to various stages of a FEL. Through meticulous analysis, the operation of the model when presented with physical data inputs is thoroughly examined. The resulting point clouds are visualized, and a detailed explanation is provided regarding the functionality of the conditionality feature. By dissecting the outcomes, this chapter sheds light on the efficacy and intricacies of the surrogate model, particularly in the context of real-world application scenarios.

Chapter 7 encapsulates the study's outcomes, delving into a detailed analysis of the methodological approach's merits and shortcomings. By scrutinizing the findings, the chapter aims to delineate the method's strengths and weaknesses, offering a nuanced perspective. Additionally, it provides insights into potential avenues for refining and optimizing the model, thereby paving the way for further advancements in the field.

# 2 Literature Review: Generative models for point clouds

In recent years, Variational Autoencoders (VAEs) [KW22] and Generative Adversarial Networks (GANs) [Goo+14] arose as one of the most popular frameworks in modeling distribution underlying the input data. Naturally, these have been used to model 3D point clouds. Such generative models find their applications in the areas of autonomous driving and object recognition.

GANs show great promise in their successful application in learning complex data distributions. Compelling results have been shown on various forms of data including text, images, and videos. Naturally, they have been applied to process 3D information .

[Ach+18] was one of the pioneering works that tested the effectiveness of GANs on point clouds. They test it on two approaches. One, the raw point cloud is directly fed to a GAN (r-GAN). Two, instead of operating on the raw point cloud, they pass the data through a pre-trained auto encoder which is trained separately for each object class with the EMD loss function. This setup is termed as I-GAN. They conclude that the I-GAN is significantly simpler and better than the R-GAN.

[Li+18] eliminates this multi-stage training and proposes an end-to-end GAN model for point clouds. It suggests a two-fold modification to the GAN algorithm for learning to generate point clouds (PC-GAN). Initially, they combine ideas from hierarchical Bayesian modeling and implicit generative models by learning a hierarchical and interpretable sampling process. Additionally, it utilizes a 'sandwiching' reconstruction objective that combines a variant of Wasserstein GAN loss with EMD.

The disadvantage of using GAN-based generative models is that the training process is notoriously hard and unstable [Sal+16]. Moreover, GAN-based models train by optimizing metrics such as EMD directly. This leads to problems like mode collapse which in turn leads to drifting from the true data distribution.

VAEs are another popular class of generative models that show similar success as that of GANs. The main advantages of these types of generators lie in their ability to encode the information with the possibility to decode and generalize new samples.

[GWM18] is one of the early papers that used VAEs to process point clouds for 3D shape understanding and generation tasks. The proposed network represents a 3D shape as a set of locality-preserving 1D ordered lists of points at multiple resolutions allowing an

efficient feed-forward processing through 1D convolutions. It argues that this representation of point clouds improves information flow across scales, enabling the network to perform coarse-to-fine analysis leading to faster convergence during training and improved shape generation quality.

[Zam+19] is another end-to-end approach that allows simultaneous learning of latent space of representation and generates 3D point clouds. In this method, the authors propose an extended Deep Adversarial Autoencoder model to accept 3D input and output. Such models that incorporate VAE into the framework usually regularize the feature space to a normal distribution. It has been shown, that there are better prior distributions to optimize for [TW18]. Moreover, due to the necessity for defining a tractable and differentiable KL divergence, VAE allows only for a narrow set of prior distributions.

Normalizing flows (NF) are a new class of generative models that approach the task of generating by directly calculating the likelihood. PointFlow [Yan+19] uses continuous normalizing flows along with a VAE to generate 3D point clouds. By conditioning point clouds on shape distributions, this method controls the final shape of the point cloud, as well as facilitates generating novel clouds of high fidelity. [Sty+20] proposes a method that leverages discrete flow which is simpler and faster to train in comparison with Pointflow. This work postulates each cloud as a parameterized probability distribution defined by a generative neural network. While both these models apply continuous and discrete NFs to yield state-of-the-art performance on shape generation and reconstruction benchmarks, they also have limitations. Transforming a standard Gaussian using an invertible map into a complex geometry accurately requires stretching and shrinking space to be infinitely strong [Cor+21], [Din+20]. [Pos+21] proposes a generalization of NFs by introducing additional discrete latent variables. Thus shapes are composed as a product of exerts using multiple NFs. Each NF specializes in a sub-region in an unsupervised fashion. We choose this method as the base for our surrogate model.

In this thesis, the model behavior is analyzed, conditionality is introduced, and comparisons are drawn with another NF-based surrogate model. Strong model reduction is performed, and unique training protocols are tested to ensure efficient use of computational resources. Finally, the developed model is tested on physical data derived from FEL simulations, and the results are studied.

# 3 Theoretical Background

In this chapter, various theoretical concepts that are fundamental to the field of machine learning are discussed, particularly those essential for the formulation of generative models. The mathematical and statistical foundations that underpin generative modeling techniques are explored. By delving into these principles, a comprehensive framework that explains how generative models are formulated is provided. This includes the probabilistic assumptions, the structure of the models, and the algorithms used for training and inference.

## 3.1 Threshold Logic Units

A threshold logic unit [Roj96] is a basic processing unit for real-valued numbers with $n$ inputs $x_1, \ldots, x_n$ and one output $y$. The unit has a threshold $\theta$. Each input $x_i$ is assigned a weight $w_i$. A threshold logic unit calculates the function:

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

The inputs are typically organized into an input vector $\mathbf{x} = (x_1, \ldots, x_n)$ and the weights into a weight vector $\mathbf{w} = (w_1, \ldots, w_n)$. Utilizing the scalar product, the condition tested by a threshold logic unit can also be expressed as $\mathbf{w} \cdot \mathbf{x} \geq \theta$.

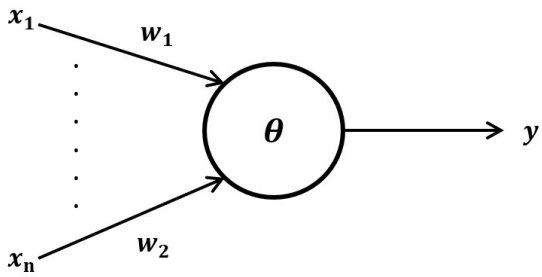


Figure 3.1: Representation of a Threshold Logic Unit

A representation of the threshold logic unit is shown in the above Fig. [3.1]. Every input is given by an arrow that points to the unit along with the weight of the associated input. Here in the unit, the threshold $\theta$ is stored. The computed output is given by $y$. This computation is iterated and the weights $w_i$ are updated until the error is minimized. Threshold logic units show great versatility when they are joined together to form a network. These networks, whose parameters are trainable, are widely used to estimate functions that can depend on several inputs and are not generally known.

## 3.2 Multilayered Perceptrons (MLP)

An artificial neural network (ANN) [MP43] consists of a series of stacked layers with each layer containing threshold logic units as the basic building blocks. These units interact with each other through weights. An MLP [Hay94] is a type of feedforward artificial neural network with multiple layers. These networks have at least three layers of threshold logic units such that every unit in a layer is connected to all the units in the following layer to form a network. Such a configuration of the network is called a fully connected neural network.

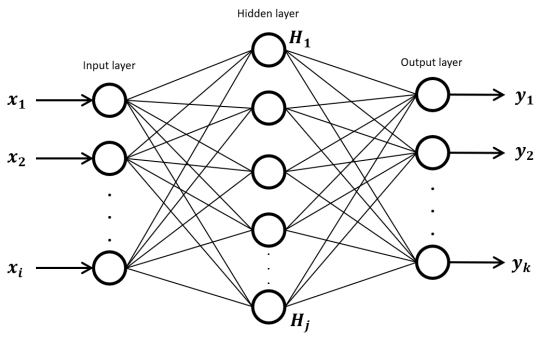


Figure 3.2: Representation of an MLP

The input nodes $X$ make up the first layer, followed by hidden layers $H$ and finally the output layers $Y$ give the processed output. The nodes in each stage compute the output based on the threshold logic unit-mapping. The weights associated with all the connections are updated by computing their gradients with respect to the error. This process is called backpropagation. MLPs can have multiple hidden layers implying that these networks can get arbitrarily deep. Such instances of MLPs form an integral part of generative modeling as they are crucial in learning functions that help better describe datasets.

## 3.3 Autoencoders

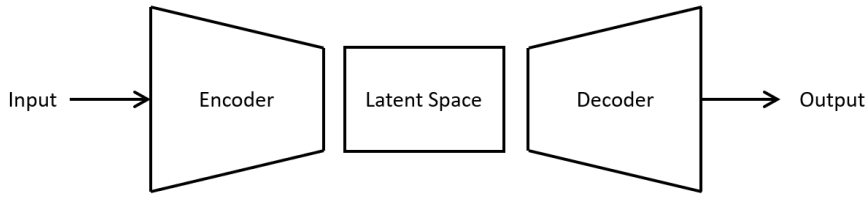


Figure 3.3: Representation of an Autoencoder Network

Autoencoders (AE) [HS06] are neural networks that consist of two parts. An encoder network that compresses high-dimensional input data into a lower-dimensional representation vector, and a decoder network that decompresses a given representation vector back to the input domain. This process is represented by Fig. [3.3]. Both these networks are trained to find the weights for the encoder-decoder pair such that the loss function between the original input and the reconstructed input is minimal. Autoencoders find their applications in data compression. However, the architecture provides a basis for a generative model called the Variational Autoencoders

## 3.4 Variational Autoencoders

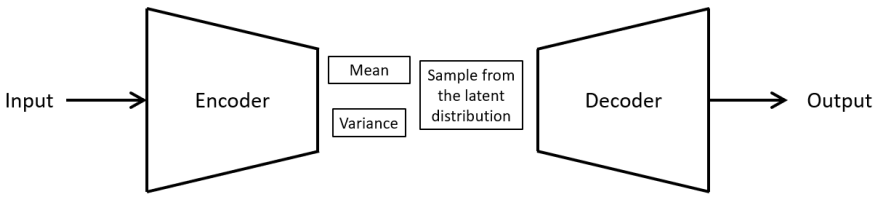


Figure 3.4: Representation of a Variational Autoencoder Network

Variational Autoencoders (VAE) [KW22] form the most fundamental and widely popular class of generative model. Although VAEs are very similar to autoencoders in the structure, they need changes in the encoder and the loss function to make the model generative. During the encoding process, each input data point is mapped to a multivariate normal distribution around a point in the latent space instead of a single point as it happens in the autoencoder. VAEs assume no correlation between any of the dimensions in the latent space and consequently, the covariance matrix is diagonal. The encoder only needs to map every point to a mean vector and a variance vector and does not need to factor in the covariances between the dimensions. Additionally, The loss function requires an additional component in the form of Kullback-Liebler (KL) divergence. This term quantifies the difference between two probability distributions. In this case, it measures how different the distribution that is given by the encoder is from a standard normal distribution. This modification ensures that the latent space is regulated and makes it easier to sample from. Adding this to the

reconstruction loss term in the loss function gives the objective function that needs to be minimized to get the appropriate weights and biases of the network.

$$\mathbb{E}_{z\sim p(z|x)}\left[-\log\left(p_\theta(x \mid z)\right)\right] + D_{KL}\left(p_\Phi(z|x)||\mathcal{N}(0,1)\right) = \mathcal{L}_D + \mathcal{L}_{\text{prior}} \tag{3.1}$$

In Eq. (3.1), $x$ represents the input, and $z$ is the latent space variable. and $\theta$ and $\Phi$ are encoder and decoder parameters respectively. The $\mathcal{L}_D$ part denotes the reconstruction error and the $\mathcal{L}_{\text{prior}}$ quantifies how different the distribution is from a standard normal distribution. This forms an adversarial setup during training such that the estimated distribution is just complex enough to represent the target data distribution with some acceptable amount of error. At the same time, it maintains the ease of sampling of a standard normal distribution.

## 3.5 Normalizing Flows

Normalizing flows (NF) [RM16] belong to a category of models that utilize a sequence of invertible functions (also referred to as bijective functions) to approximate an unknown probability density function $p(x)$. The concept behind normalizing flows involves understanding the relationship between a desired distribution $p(x)$ and a standard normal distribution $p(y)$ through functions. Subsequently, samples of the desired distribution are generated by inputting samples from Gaussian prior during inference into these reversible functions. Neural networks are used to estimate these functions between $X$ and $Y$ utilizing log-likelihood as the loss function, for training the networks. A visual representation illustrating the concept of flows is depicted in Fig. [3.5].

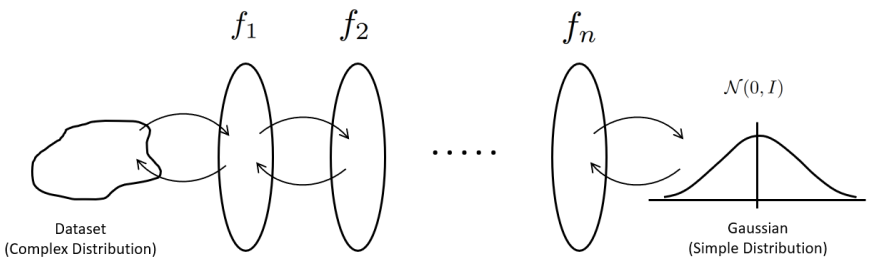


Figure 3.5: Representation of Normalizing flows

By the change of variable formula, the unknown probability distribution $p(x)$ can be expressed in terms of the standard normal as follows:

$$p(x) = p(y)\left|\det\left(\frac{\partial f(y)}{\partial y^T}\right)\right|^{-1} \tag{3.2}$$

where $f$ is the function that maps inputs from $x$ to $y$ and denotes the $n$ invertible functions in sequence. Such models that leverage the invertibility and learn the mappings are called flow-based models. The parameters of the functions that constitute these models are estimated by minimizing the negative log-likelihood that can be derived based on Eq. (3.2) and the loss function would be

$$\mathcal{L}(\theta) = \mathbb{E}_{x\sim p(x)}\left[-\log\left(p_f(x)\right)\right] = -\mathbb{E}_{x\sim p(x)}\left[\log(p(y)) - \sum_{i=0}^{n-1}\log\left(\det\left|J_{f_i^{-1}}(x)\right|\right)\right] \tag{3.3}$$

In practical applications, computing the Jacobian in Eq. (3.3) is computationally expensive, especially considering that neural networks are not inherently invertible. A method to address this challenge is by employing flow-based models, which simplify Jacobian computations and ensure invertibility, such as the Real Non-Volume Preserving coupling block [DSB17] (RNVP coupling block). In this approach, input data is partitioned into two segments. One segment undergoes an affine transformation while the other remains unchanged. This strategy results in a Jacobian matrix that is upper triangular, significantly reducing the computational complexity associated with computing its determinant. The affine transformation is executed by two arbitrary functions in the form of neural networks, which provide the shift and scale values. While neural networks alone cannot perform inverse transformations, the RNVP coupling block, illustrated in Fig. [3.6], depicts the inversion of the affine transformation.

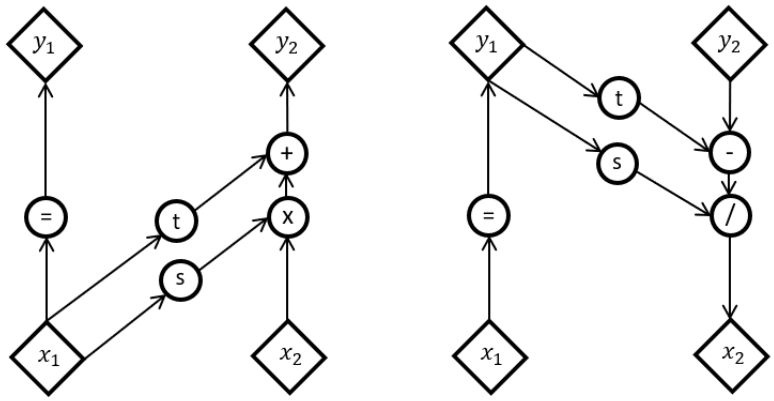


Figure 3.6: Real NVP coupling block architecture [DSB17]

The coupling layer involves a straightforward scale and shift operation applied to a subset of variables within the current layer, with the remaining variables used to compute the scale and shift. Consequently, the inversion operation becomes simple as the shift and scale factors remain consistent. [DSB17] provides a detailed mathematical description of the RNVP coupling blocks. An enhanced version of these RNVP coupling blocks, emphasizing controllability, is the conditional affine coupling block [Ard+19], which introduces conditioning functionality. This extension enables the model to learn from and generate categorical data. The architectural modification is depicted in Fig. [3.7].

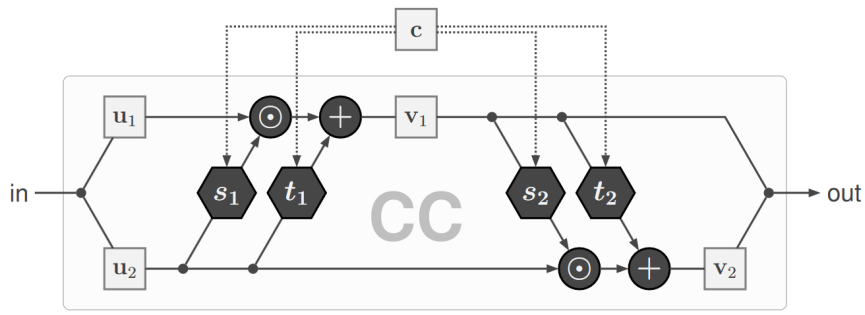


Figure 3.7: Architectural representation conditional Invertible Neural Network [Ard+19] (cINN)

The design of the coupling block depicted in Fig. [3.7] yields a conditional variant of the coupling block. Since the sub-networks $s_1$, $s_2$, $t_1$, and $t_2$ are not required to be inverted, the conditioning data $c$ is concatenated to the inputs of these sub-networks, preserving invertibility.

## 3.6 Mixture of Normalizing flows

The 'Mixture of Normalizing Flows' model [Pos+21] represents a modification of a standard VAE where the prior distribution $p(z)$ is typically assumed to be a standard normal distribution for learning the decoder distribution $p_\theta(x|z)$. As an enhancement to this standard VAE framework, the Mixture of NFs model learns the prior distribution $p(z)$ by utilizing an unconditional version of normalizing flows. Moreover, the decoder distribution $p_\theta(x|z)$ is learned through $m$ conditional normalizing flows. This constitutes a fundamental aspect of the model, as it aims to address the challenge of complex geometries by capturing different segments of the point cloud using distinct normalizing flows. A schematic representation of the training and generation processes is illustrated in Fig. [3.8].

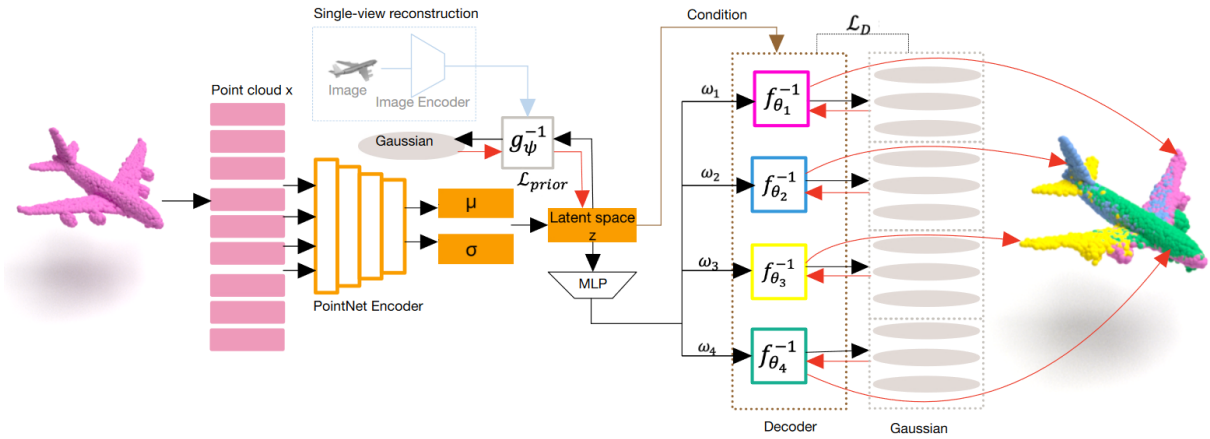


Figure 3.8: Schematic representation of the architecture mixture of normalizing flows from [Pos+21]

The black arrows in Fig. [3.8] illustrate the direction of the training process, while the red arrows represent the generation process. The decoder component comprises $m = 4$ conditional RNVP coupling blocks, responsible for learning and generating the decoder

distribution. For better understanding, the model can be divided into these four parts based on their functionality.

1. **PointNet Encoder ($f_{PNE}$)**: PointNet architecture **[Qi+17]** provides a framework that can be applied in a variety of point cloud processing such as object detection, classification and segmentation. In this work, the permutation invariant of PointNet is used by omitting the segmentation and classification networks. This initial step in the model directly processes point clouds. Permutation invariance implies that the encoded representation remains unchanged regardless of the input point sequence. During training, this encodes the input data into a higher-dimensional feature space and plays no role in generation. Notationally, it is represented as

$$f_{PNE} : \mathbb{R}^{d \times n}_{\text{input}} \longrightarrow \mathbb{R}^{P \times n}_{\text{feature}}$$

   where $d$ represents the dimensionality of the input points, $n$ is the number of points, and $P$ is the dimensionality of the feature space determined by the last layer of the PointNet encoder.
2. **Prior Flow ($g_{\psi}^{-1}$)**: This represents an unconditional version of an RNVP flow, sharing the same architecture as shown in **Fig. [3.6]**. It accepts the latent representation of the point cloud and maps it to a Gaussian distribution during training, and performs the inverse during generation. Mathematically, it is expressed as

$${g_{\psi}}^{-1} : \mathbb{R}^{g}_{\text{posterior}} \longrightarrow \mathbb{R}^{g}_{\text{Gaussian}}$$

   where $g$ denotes the size of the latent space.
3. **Weights Encoder ($f_w$)**: This MLP block comprises a simple feed-forward network that takes samples from the latent space distribution and assigns them to $m$ normalizing flows in the decoder distribution. These weights are optimized by the optimizer to minimize the overall model loss. The weights represent the probabilities that each decoder flow operates on a given input data point, with the constraint that the sum of $m$ weights is 1.

$$f_w : \mathbb{R}^{g} \longrightarrow \mathbb{R}^{m}_{\text{weights}} \text{ such that } \sum_{i=1}^{m} w_i = 1$$

4. **Decoder Block ($f_\theta$)**: This block comprises $m$ conditional RNVP flows. Based on the weights assigned to these flows, they map the samples of input point clouds to samples of normal distribution during training, and vice versa during generation. Regardless of the process, they are always conditioned by samples from the latent space.

$$f_\theta : \mathbb{R}^{d \times q}_{\text{input}} \longrightarrow \mathbb{R}^{d \times q}_{\text{Gaussian}}$$

   Here $q$ represents a fraction of the regenerated point cloud by each decoder such that $\sum_{i=1}^{m} q_i = n$.

### 3.6.1 Loss Function

The formulation of the loss function is pivotal for the success of a machine learning algorithm. The Mixture of NFs model integrates standard architectures of VAEs and strategically employs

normalizing flows. Therefore, the design of the loss function should encompass all these sub-parts. Since the decoder distribution $p_\theta(x|z)$ is a linear combination of $m$ normalizing flows, substituting Eq. (3.3) into $p(x|z)$ of Eq. (3.1) yields

$$\mathcal{L}_D = \mathop{\mathbb{E}}_{z\sim p_\phi}\left[-\log\left(\sum_{i=0}^{m} w_i \mathcal{N}\left(f_{\theta_i}^{-1}(x);\mu(z),\Sigma(z)\right) * \det\left|J_{f_{\theta_i}^{-1}}(x;z)\right|\right)\right] \tag{3.4}$$

where $w_i$ are weights (scalars) calculated by the multi-layered perceptrons ($f_w$) based on samples from the latent space $z$. These weights correspond to $m$ decoder flows. The terms $\mu(z)$ and $\Sigma(z)$ give the mean and covariance of $\mathcal{N}\left(f_{\theta_i}^{-1}(x)\right)$, the hyperparameters of the model. The function $f_{\theta_i}^{-1}$ signifies the inverse transform from a Gaussian distribution to $p_\phi(z|x)$ . This constitutes the $\mathcal{L}_\mathcal{D}$ component of Eq. (3.1).

In contrast to a standard VAE discussed in section 3.4, the prior distribution $p(z)$ here is a learned distribution with parameters $\psi$. Hence, equation $L_{\text{prior}}$ from Eq. (3.1) can be rewritten as:

$$\mathcal{L}_{\text{prior}} = -\mathcal{H}\left(p_\phi\right) - \mathop{\mathbb{E}}_{z\sim p_\phi(z|X)}\left[\log\left(p_\psi(z)\right)\right] \tag{3.5}$$

where:

$$\mathcal{H}\left(p_\phi\right) = \sum_{z\sim p_\phi}\left(p_\phi \log\left(p_\phi\right)\right) \tag{3.6}$$

If $\mathbb{E}\left[\log\left(p_\psi(z)\right)\right]$ is denoted as $GNLL$ (Gaussian Negative Log-Likelihood) and $\mathcal{H}\left(p_\phi\right)$ as $GENT$ (Gaussian Entropy), and they are scaled with weights instead of being summed as seen in Eq. (3.1), the loss function can be rewritten as:

$$loss = c_{\mathcal{L}_D} \cdot \mathcal{L}_D + c_{GENT} \cdot GNLL + c_{GENT} \cdot GENT \tag{3.7}$$

where $c_{\mathcal{L}_D}$, $c_{GENT}$, and $c_{GENT}$ are associated weights, all positive real numbers. These weights determine the contributions of the learned decoder loss, the negative log-likelihood of the prior distribution, and the entropy of the prior distribution, respectively. Deconstructing the loss function in this manner facilitates better training of the neural networks.

## 3.7 Sinkhorn Distance

Sinkhorn distance [Cut13], also known as entropic regularized optimal transport, is a variant of the optimal transport problem that incorporates an entropy regularization term, enhancing computational efficiency and numerical stability.

### 3.7.1 Optimal Transport Problem

The optimal transport problem aims to find the most efficient way to transform one probability distribution $\mu$ into another $\nu$, minimizing a given cost function $c(x, y)$. The objective

is to determine a transport plan $\gamma$ that minimizes the total cost:

$$\min_{\gamma \in \Pi(\mu,\nu)} \int_{X \times Y} c(x,y)\, d\gamma(x,y)$$

where $\Pi(\mu,\nu)$ is the set of all transport plans that have $\mu$ and $\nu$ as marginals.

### 3.7.2 Entropy Regularization

To make the optimal transport problem more tractable, an entropy term is added to the objective function:

$$\min_{\gamma \in \Pi(\mu,\nu)} \left[ \int_{X \times Y} c(x,y)\, d\gamma(x,y) + \varepsilon H(\gamma) \right]$$

where $H(\gamma)$ is the entropy of the transport plan $\gamma$:

$$H(\gamma) = -\int_{X \times Y} \gamma(x,y) \log \gamma(x,y)\, dx\, dy$$

and $\varepsilon$ is a regularization parameter controlling the balance between the transport cost and entropy.

### 3.7.3 Sinkhorn Algorithm

The Sinkhorn algorithm is an iterative method used to solve this entropic regularized optimal transport problem. It alternates between updating scaling factors for rows and columns of the transport plan matrix, ensuring the marginals match the distributions $\mu$ and $\nu$. This process involves:

1. Computing the element-wise exponential of the cost matrix scaled by the regularization parameter.
2. Normalizing steps to enforce the marginal constraints.

Sinkhorn Distance is extensively used in this project to measure the difference in the probability distributions of the input point clouds and the point clouds generated by the data-driven surrogate model.

# 4 Experimentation on Mixture of NFs

In this chapter, a comprehensive sequence of experiments is presented, aimed at refining the model architecture to better align with the requirements of a data-driven surrogate model. The initial focus is on examining the model's adaptability to various point cloud geometries. The model is tested on point clouds with distinct features to ensure that the surrogate model can accurately learn and generate point clouds with complex geometries.

Following this, the introduction of conditionality to the model is explored by utilizing the single-view reconstruction feature. This conditionality is introduced in two forms i.e. scalar conditioning and one-hot vector conditioning. The ability to generate different configurations of point clouds according to classes set by the condition is imparted to the surrogate model by conditioning the prior distribution.

Additionally, the model's reconstruction performance is assessed across various resource utilization measures and variations in input data characteristics, such as the local densities of the point clouds and their dimensionality. These results are compared to those obtained from a single normalizing flow-based model to evaluate improvements and identify potential areas for enhancement.

Subsequently, strategies aimed at reducing the number of trainable parameters and the overall training duration are outlined. This involves a detailed discussion on optimizing the model to be both efficient and effective. Finally, the behavior of conditionality is scrutinized in the last section, providing insights into how these mechanisms impact the model's performance and accuracy.

At the end of this chapter, a well-rounded analysis of the surrogate model's capabilities, enhancements through conditionality, and efficiency in terms of resource utilization and training optimization is presented.

## 4.1 Model behavior on simple geometries as input data

The Mixture of NFs model is trained on point clouds exhibiting a range of geometric attributes. It aims to generate point clouds with diverse geometric characteristics, reflecting those commonly observed in plasma physics. One of such geometries is that of a torus. In this

experiment, the input point cloud lacks geometric irregularities like sharp corners. The target point cloud's distribution forms a smooth, continuous shape. Throughout the training, the model performance in terms of reconstruction is observed along with the weights.

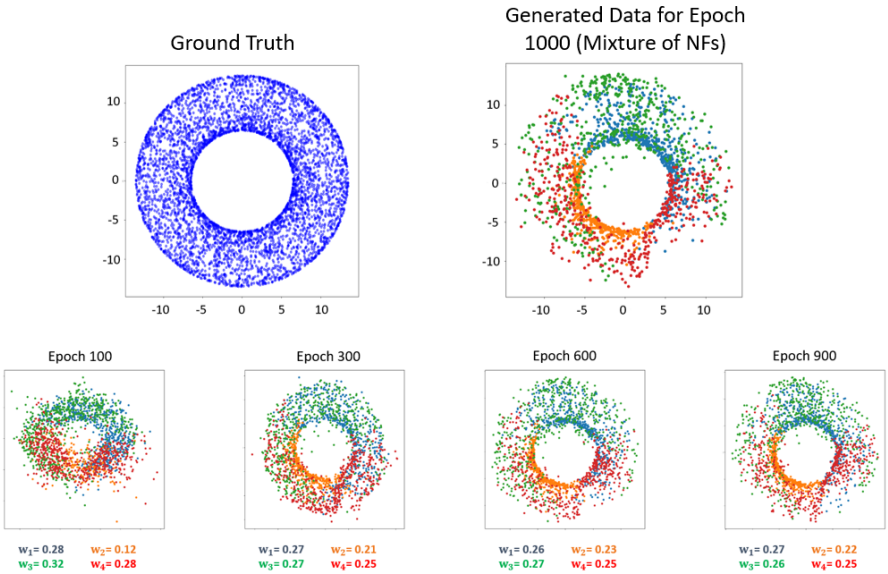


Figure 4.1: Ground Truth vs Generated data of a torus-shaped point cloud

From Fig. [4.1], it becomes evident that the decoders specialize in local clusters without any recognizable geometric characteristics due to the absence of sharp geometric transitions. The weights assigned to the decoders reflect this observation, as all four decoders exhibit similar weights, suggesting they contribute equally to generating the point cloud of the target distribution.

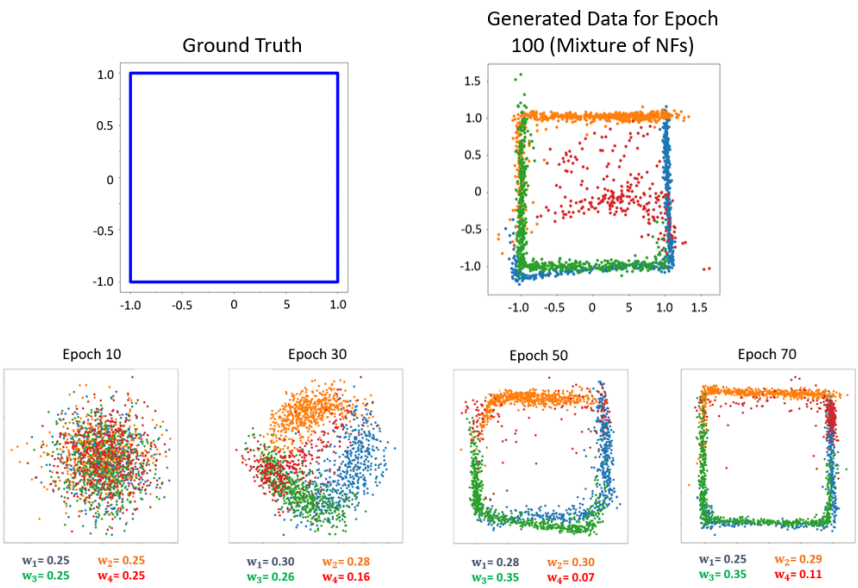


Figure 4.2: Ground Truth vs Generated data of a square-shaped point cloud

For a two-dimensional square input, the decoders become adept at specializing by separating at points of irregularities as shown in Fig. [4.2]. It's also important to note that the model

regulates the weights assigned to each decoder. For example, $\mathbf{w_4}$ has the lowest value because it represents a region (seen in red) that deviates from the ground truth at the 100th epoch.

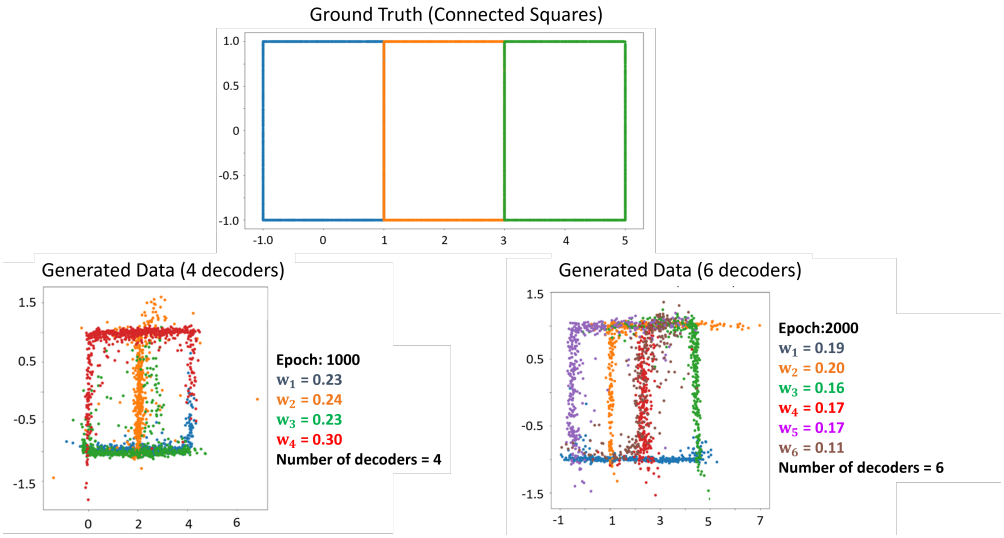


Figure 4.3: Ground Truth vs Generated data point cloud that is in the shape of connected squares

To study the impact of the number of decoders, a 2D geometry consisting of three interconnected squares is used as the input data, and the resulting generated data is examined. It becomes apparent from Fig. [4.3] that the model employing four decoders struggles to capture all three squares, whereas the model with six decoders yields improved outcomes. This underscores the significance of selecting an appropriate number of decoders to produce geometrically precise point clouds.

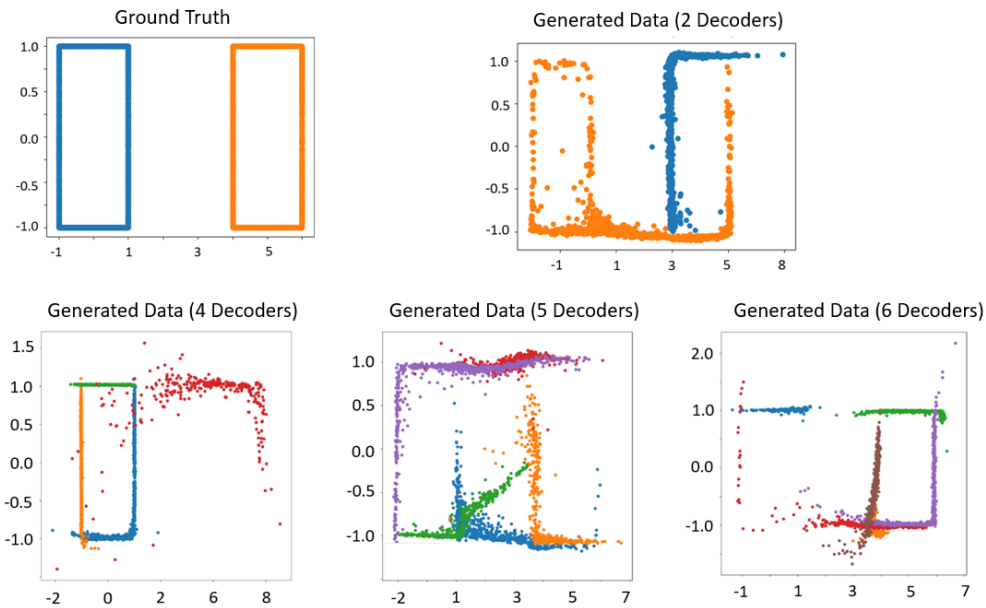


Figure 4.4: Ground Truth vs Generated data point cloud that is in the shape of disconnected squares

In instances of disconnected geometries, as depicted in Fig. [4.4], it is noted that the generated point cloud consistently retains continuity. This remains true irrespective of the number of decoders employed.

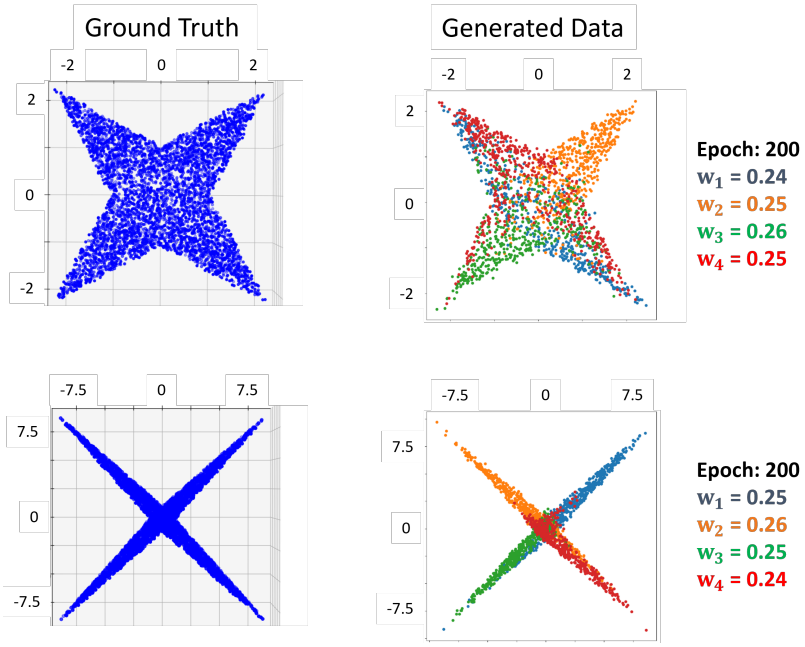


Figure 4.5: Ground Truth vs Generated data point cloud that is in the shape of stars. The model learns to specialize in different arms of the stars as this length increases

In plasma physics, point clouds often exhibit slender features. These features pose a challenge as flow-based models need to compress a standard normal distribution into shapes with high aspect ratios. To assess the efficacy of the Mixture of NFs model, star-shaped point clouds of varying arm lengths are formulated as the input datasets, and model behavior is assessed. As illustrated in Fig. [4.5], it is evident that as the length of the arms increases, the Mixture of NFs model learns to specialize in distinct arms of the star. This characteristic of the model holds promise for improved reconstruction of point clouds with complex shapes.

## 4.2 Architectural Modification for conditionality

As discussed in the chapter [1] the existing architecture of the Mixture of NFs lacks the capability for conditional learning, hindering the model's ability to learn and generate diverse point clouds based on conditions. The architecture uses Single-View Reconstruction to condition the prior distribution on a latent representation of an image produced by ResNet18 [He+15] as shown in Fig. [4.6]. This feature of the model is utilized to introduce conditionality into the model by replacing the SVR component with a representation of the condition

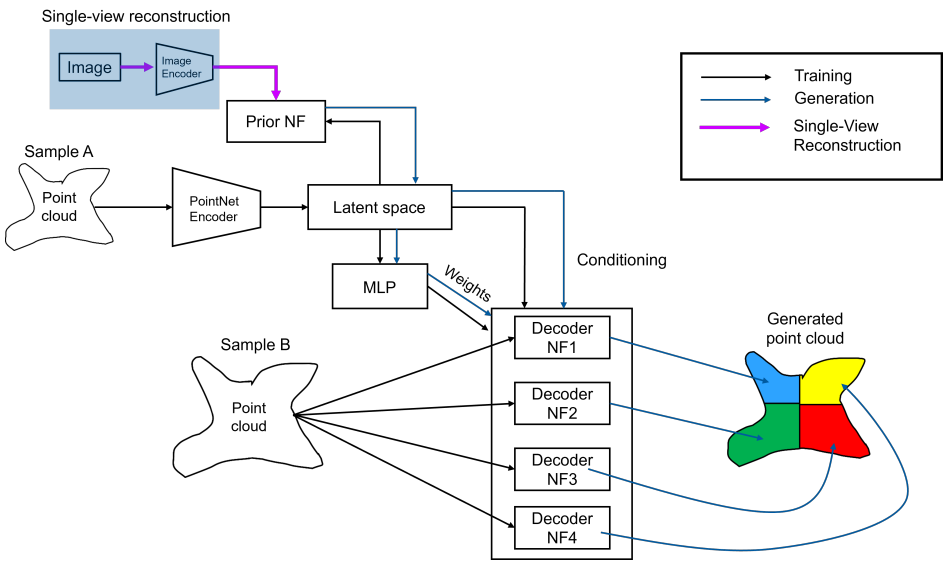


Figure 4.6: Schematic representation of Single-View Reconstruction (SVR) by the architecture of Mixture of NFs

As shown in Fig. [4.7], the prior NF can be reformulated as a conditional NF. In this approach, the prior learning block ($\mathbf{g}_\psi$) is conditioned on the categorical data representing the input point cloud, as depicted in Fig. [4.7]. This method generates different prior distributions based on the various conditions. These condition-specific prior distributions are then utilized in the decoder block to generate diverse point clouds that adhere to the given conditions. Both models are trained, and their results are analyzed to evaluate the effectiveness of this conditional approach.

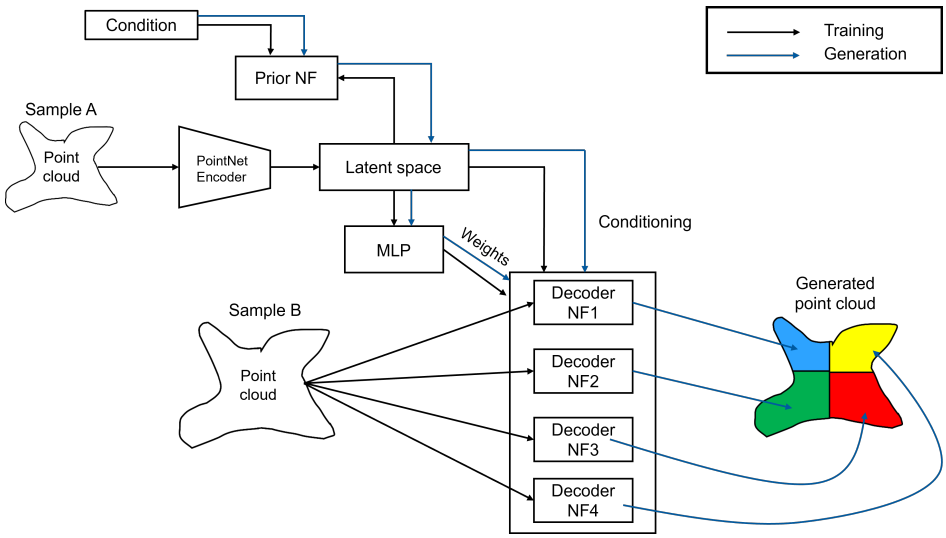


Figure 4.7: Adaptation of Mixture of NF by replacing the image input with conditional data

To check the effectiveness of this adaptation, an input dataset featuring four distinct conditions, each representing a different star configuration, is synthesized. These conditions are represented through two types of conditioning: scalar and one-hot vector. In the one-hot vector representation, the model is conditioned with a strict categorization of the various star configurations. Consequently, generating point clouds that bridge between distinct conditions becomes challenging. Conversely, scalar conditioning facilitates the inference

of intermediate point cloud configurations between any two given conditions, albeit with a trade-off in generation quality.

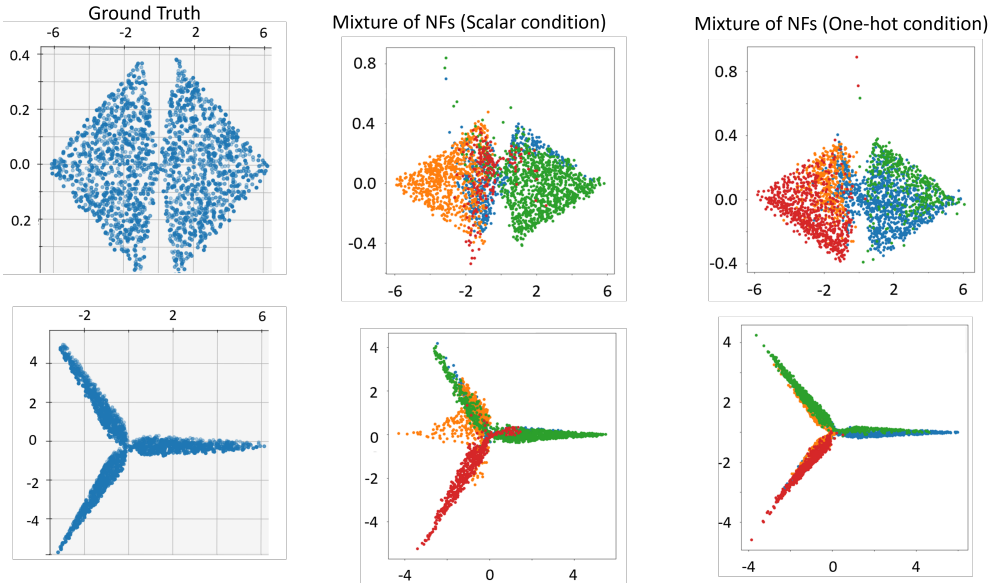


Figure 4.8: The generated data is compared with the ground truth (left). Scalar conditioned point clouds (middle) and one-hot vector conditioned point clouds (right) are generated by multiple NFs specializing in different regions

The images in Fig. [4.8] show the outcomes of these modifications with scalar and one-hot vector conditioning. It is observed that the replacement of the Single-View Reconstruction block with the conditional information equips the model with the feature of conditional generation. Furthermore, both the forms of conditioning, scalar and one-hot vector forms, effectively generate different configurations of the point clouds that correspond to the condition.

## 4.3 Comparative Analysis: Mixture of NFs versus Single NF-based Model

These experiments compare the performance of the conditional Mixture of Normalizing Flows (NFs) model with the conditional single NFs-based model on the [Ard+18] dataset. The comparison focuses on metrics such as reconstruction quality, response to increasing input data dimensions, and sensitivity to density fluctuations, all while maintaining a similar network size for both these models. Single NF-based models transform a standard Gaussian using an invertible map into a complex geometry that requires squeezing/stretching the space infinitely strongly. The Mixture of NFs on the other hand generalizes NFs by introducing additional discrete latent variables which make the point cloud shape a composition of product of experts using multiple NFs. This approach provides better reconstruction performance and tackles the problem of mismatching abstract geometric properties of the source and target distributions by patching the parts together by using multiple invertible maps. Additionally, using a single NF to transform a standard Gaussian into complicated

geometries requires infinite bi-Lipschitz constants and to achieve such constants, it is necessary to use large and deep NFs which makes them impractical.

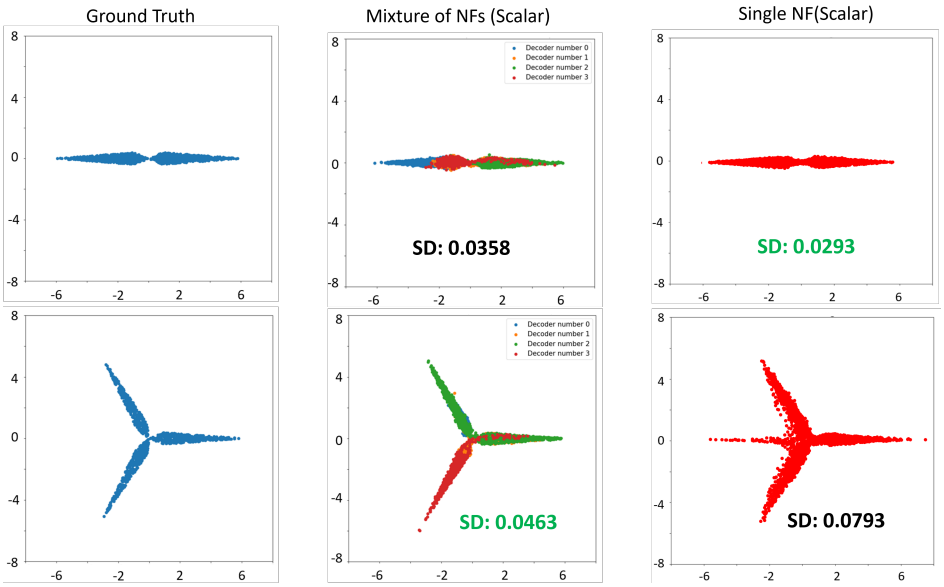


Figure 4.9: Generated data with scalar conditioning and corresponding Sinkhorn distance (SD) values

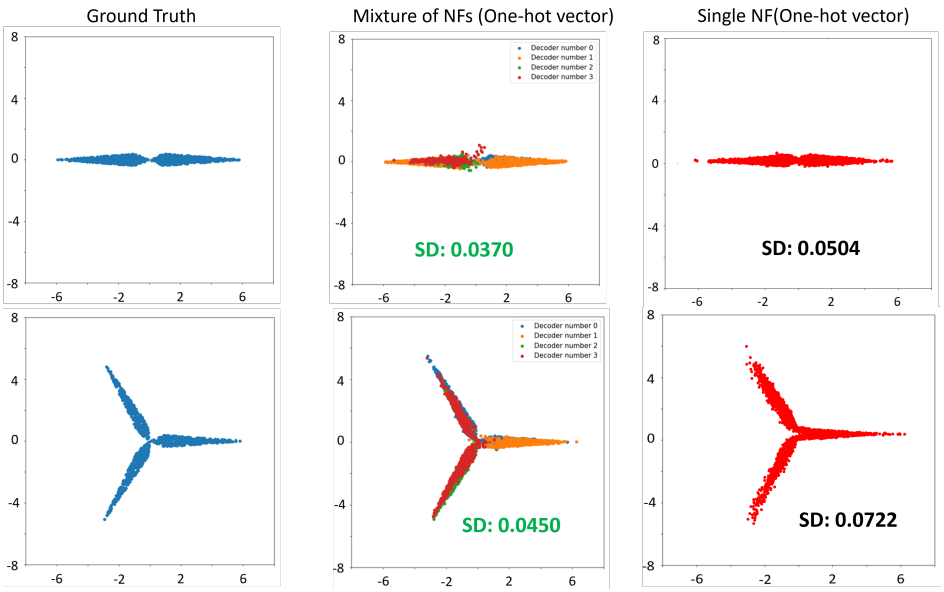


Figure 4.10: Generated data with one-hot vector conditioningand corresponding Sinkhorn distance (SD) values

Fig. [4.9] and Fig. [4.10] depict the generated point clouds corresponding to different conditions alongside the Sinkhorn distance measurements. It is noted that, for both models, the Sinkhorn distance value increases with the complexity of the generated point cloud as seen in Tab. [4.1], indicating a deterioration in the quality of the generated point clouds. Qualitative analysis of the aforementioned figures reveals that this issue is addressed by the Mixture of the NFs model, which employs different NFs to specialize in distinct regions. Although the single NF model performs slightly better than the Mixture of NFs model, according to the tabulated values, in a few cases, the former model lacks the mechanism

for reconstructing complex point clouds. This behavior can be observed by comparing the scalar conditioning for 5-armed stars both these models.

Table 4.1: Comparison of the minimum Sinkhorn distances for point clouds generated by the Mixture of NFs model and the Single NF model across 25 samples, highlighting differences in reconstruction based on conditioning types.

| Conditions | Mix of NFs | | Single NF | |
|---|---|---|---|---|
| | scalar | one-hot vector | scalar | one-hot vector |
| 2-armed stars | 0.0358 | 0.0370 | 0.0293 | 0.0504 |
| 3-armed stars | 0.0463 | 0.0450 | 0.0793 | 0.0722 |
| 4-armed stars | 0.0681 | 0.0634 | 0.0663 | 0.0512 |
| 5-armed stars | 0.0823 | 0.0969 | 0.1308 | 0.0868 |

In the scenario of scalar conditioning, it is observed that the single NF model exhibits a gradual transition in the configurations of the point clouds. In contrast, the mixture of normalizing flows demonstrates a sudden change in the generated point cloud configuration, as depicted in Fig. [4.11]. This discrepancy arises from the fact that the decoder of the Mixture of NFs is conditioned on a prior sample, which is already an output of a normalizing flow block. Conversely, the single NF block utilizes only a single stage of conditioning, resulting in a gradual evolution of the star configurations for intermediate values. This represents one of the limitations of employing the mixture of NFs model.

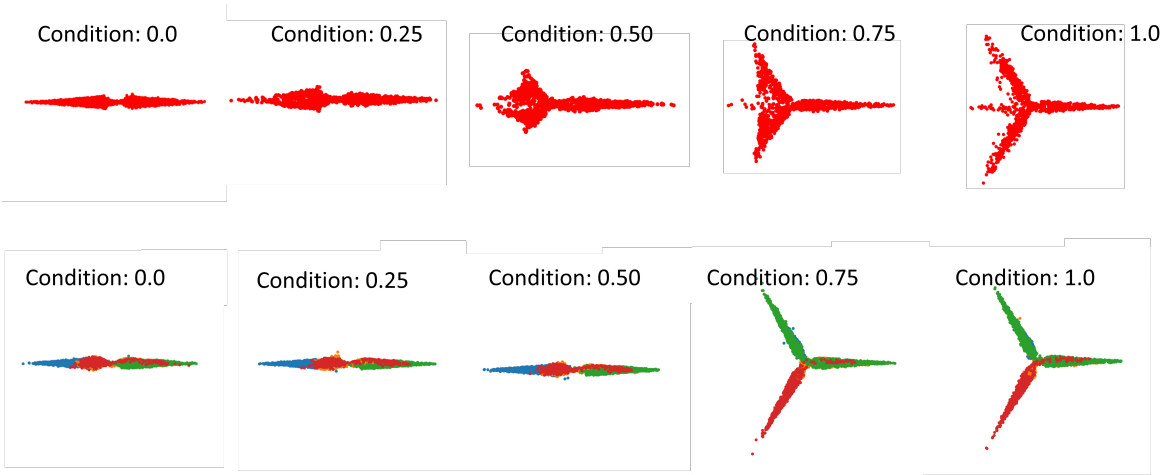


Figure 4.11: Generated data for intermediate values of condition for scalar conditioning

Considering the enhanced sophistication offered by the mixture of NFs model, it is imperative to assess the computational overhead it entails. Three key parameters are examined for this purpose: the number of trainable parameters, training time, and inference time. These metrics are summarized in Tab. [4.2].

Table 4.2: Resource Consumption Comparison

| Model | Trainable Parameters | Training Time (per epoch) | Inference Time (per sample) |
|---|---|---|---|
| Mix of NFs | 5.4 million | 6 sec | 0.25 sec |
| Single NF | 5.6 million | 2.5 sec | 0.053 sec |

This comparison reveals that when comparable-sized networks are used for generating point clouds, the mixture of NFs model performs generally better in terms of reconstruction of the target point cloud distributions. Single NF models generate the entire volume of the cloud single-handedly whereas each decoder in the mixture of NFs is responsible for a portion of the point cloud which aids the model in reconstructing the intricate details of the target point cloud. However, this sub-region specialization comes at a considerable processing time. Even though the single NF models have a comparable number of parameters, the training time is significantly less. This demands a strong reduction in the trainable number of parameters of the Mixture of NFs model.

### 4.3.1 Higher Dimensional Input

In plasma physics, point clouds are frequently encountered in six-dimensional phase space which includes space and momentum dimensions, underscoring the importance of assessing model behavior with higher-dimensional inputs. To investigate this, input point clouds are synthesized as 6D point clouds. The initial three dimensions encapsulate star configurations with constant thickness, while the additional three dimensions denote the same star configurations but with different rotations. The 2D projections of such a 6D star are shown in Fig. [4.12]. This dimensional augmentation is applied to all four conditions representing distinct configurations.

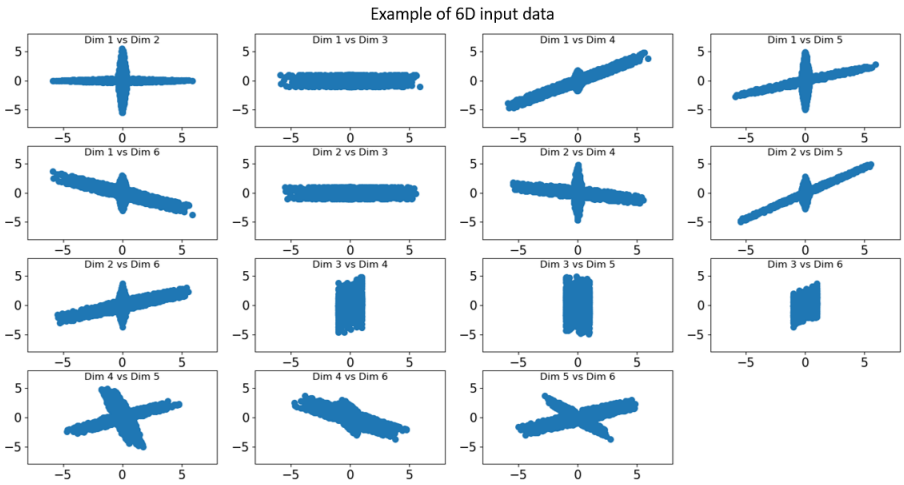


Figure 4.12: Two-dimensional projection of the six-dimensional input data

Both, the Mixture of NFs model and single NF model, are trained on this 6D data with 4 conditions for 1000 epochs for both the form of conditioning are shown in Fig. [4.13], Fig. [4.14], Fig. [4.15], and Fig. [4.14] along with the Sinkhorn distances for each case

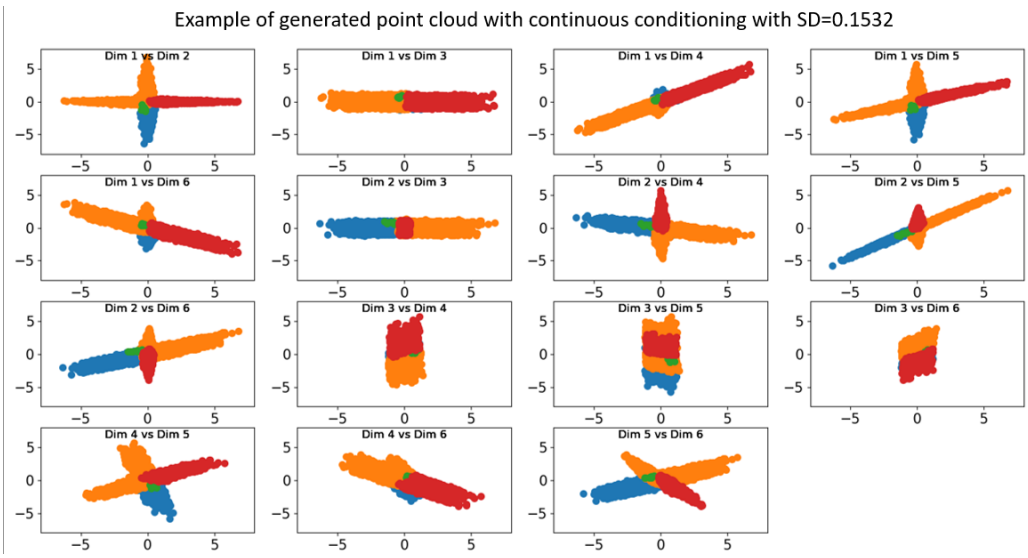


Figure 4.13: Two-dimensional projections of the six-dimensional generated data by Mixture of NFs using scalar conditioning. The different colors indicate the points generated by the respective decoders

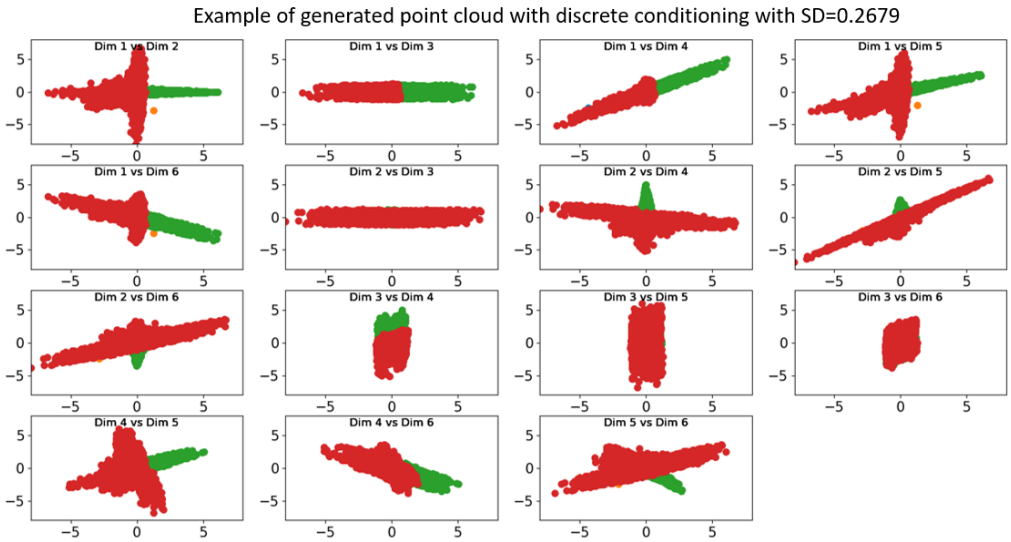


Figure 4.14: Two-dimensional projections of the six-dimensional generated data by Mixture of NFs using one-hot conditioning. The different colors indicate the points generated by the respective decoders

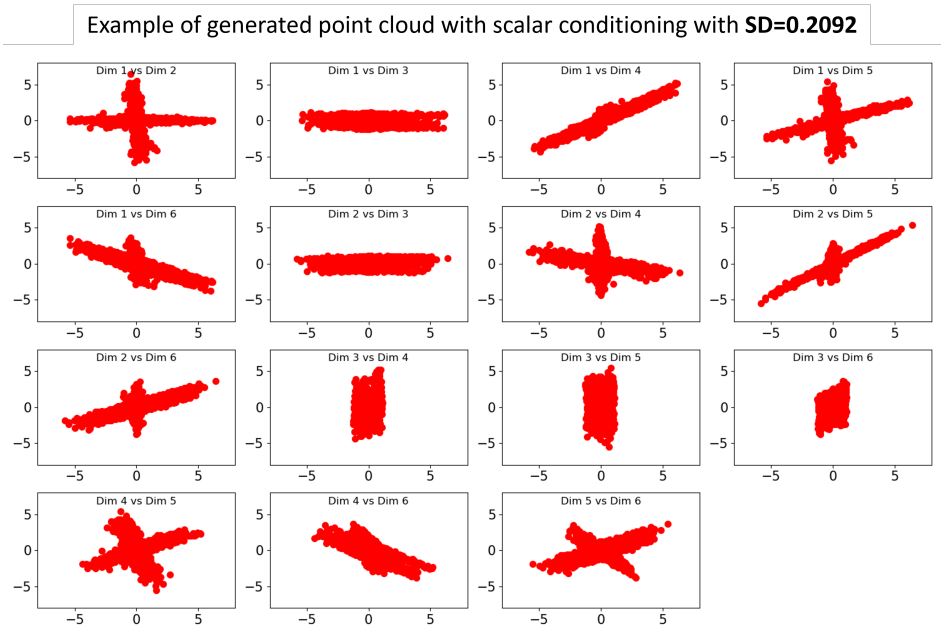


Figure 4.15: Two-dimensional projections of the six-dimensional data generated by a single normalizing flow (NF) with a similar size to that of Mixture of NF using scalar conditioning.

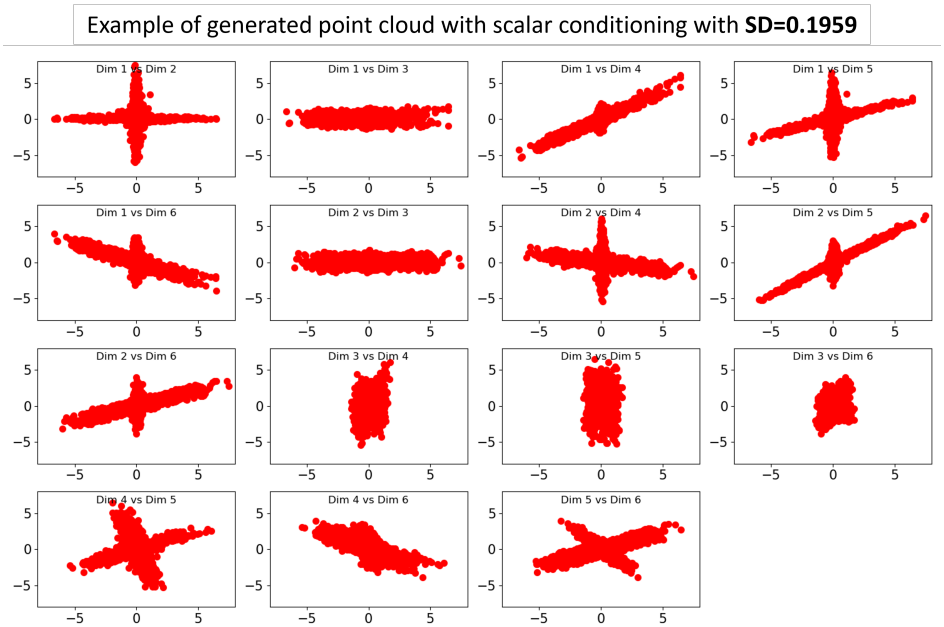


Figure 4.16: Two-dimensional projections of the six-dimensional data generated by a single normalizing flow (NF) with a similar size to that of Mixture of NF using scalar conditioning.

Table 4.3: Comparison of the minimum Sinkhorn distances for six-dimensional point clouds generated by the Mixture of NFs model and the Single NF model across 25 samples, highlighting differences in reconstruction based on conditioning types.

| Conditions | Mix of NFs | | Single NF | |
|---|---|---|---|---|
| | scalar | one-hot vector | scalar | one-hot vector |
| 2-armed star | 0.1250 | 0.1467 | 0.1041 | 0.1324 |
| 3-armed star | 0.1368 | 0.1568 | 0.1749 | 0.2108 |
| 4-armed star | 0.1534 | 0.2679 | 0.2092 | 0.1958 |
| 5-armed star | 0.2034 | 0.3454 | 0.2481 | 0.2367 |

Tab. [4.3] shows minimum Sinkhorn distances for all the conditions across 25 samples. It can be observed that the scalar form of conditioning outperforms one-hot vector conditioning in both the Mixture of Normalizing Flows (Mixture of NFs) model and its single NF counterparts. This improvement is due to sub-region specialization across six dimensions. However, the one-hot vector conditioning fails to perform on par with the scalar conditioning. This can be explained by the GNLL values for 1000 epochs as seen in Fig. [4.17]. Lower GNLL values indicate more accurate learning of the prior distribution which in turn aids a better reconstruction of the target point cloud. Meanwhile, it can be observed the Mixture of NFs model performs marginally better than the single NF model for scalar conditioning. It also learns to specialize in sub-regions when the input dimensionality scales

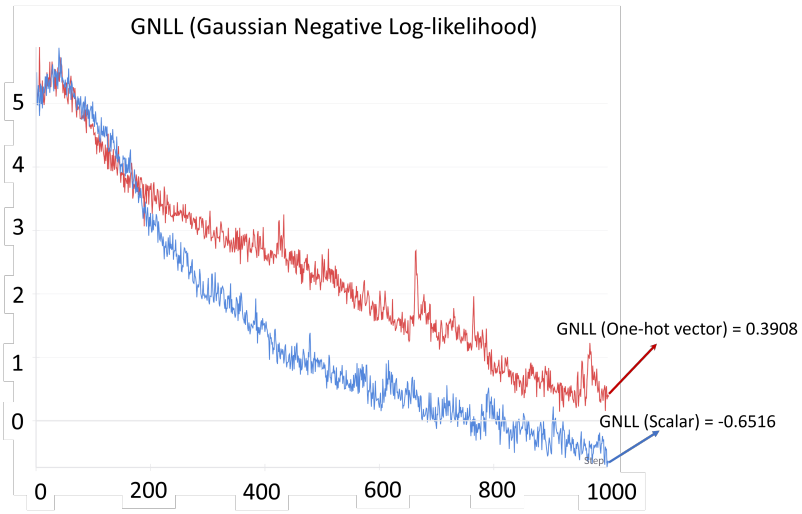


Figure 4.17: GNLL values at the end of training for scalar and one-hot vector conditioning

### 4.3.2 Variation in Density

In addition to investigating the model behavior for higher dimensional input, it is also important to check how the model preserves density variations and features of point clouds in the regions where the points are sparsely populated. To check this and compare it with the single NF model, star-shaped-point clouds that have variations in the density of the arms are used. Both the models are trained and the corresponding results are studied.

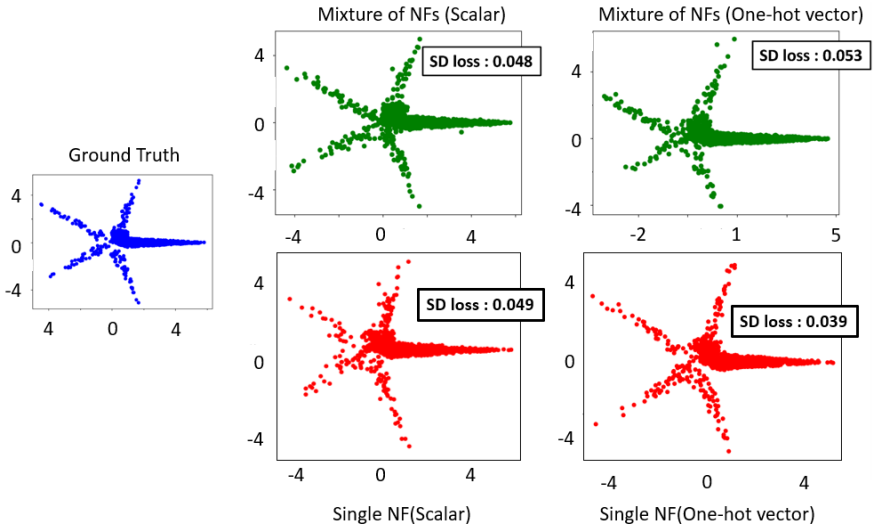


Figure 4.18: Generated data for both the models and condition forms for point clouds that vary in density. The ground truth is given by the blue point cloud while the point cloud generated by the Mixture of NFs is shown in green. The output of single NF is shown in red

Table 4.4: Comparison of the minimum Sinkhorn distances for point clouds with variation in densities generated by the Mixture of NFs model and the Single NF model across 25 samples, highlighting differences in reconstruction based on conditioning types.

| Conditions | Mix of NFs | | Single NF | |
|---|---|---|---|---|
| | scalar | one-hot vector | scalar | one-hot vector |
| 2-arm star | 0.038 | 0.032 | 0.039 | 0.029 |
| 3-arm star | 0.054 | 0.038 | 0.037 | 0.033 |
| 4-arm star | 0.048 | 0.043 | 0.047 | 0.039 |
| 5-arm star | 0.048 | 0.053 | 0.049 | 0.039 |

As seen in Fig. [4.18], both the models learn to correctly represent the densities. The mixture of NFs model with scalar conditioning learns to capture the feature with a lower number of points in comparison with the single NF model. Minimum Sinkhorn distances across 25 samples for each condition are shown in Tab. [4.4] suggesting that the reconstruction quality is alike across conditions.

## 4.4 Model behavior with increased number of conditions

In this section (4.4), the model behavior when the number of conditions is increased is studied. Starting with 4 distinct point clouds, the number of conditions is expanded to 24. Each star is rotated by multiples of 60 degrees to create a total of 24 different star configurations, which are then used as input to the Mixture of NFs model with scalar conditioning. Multiple samples are drawn for each condition, and it is observed that the generated point clouds do not always conform to the given condition.

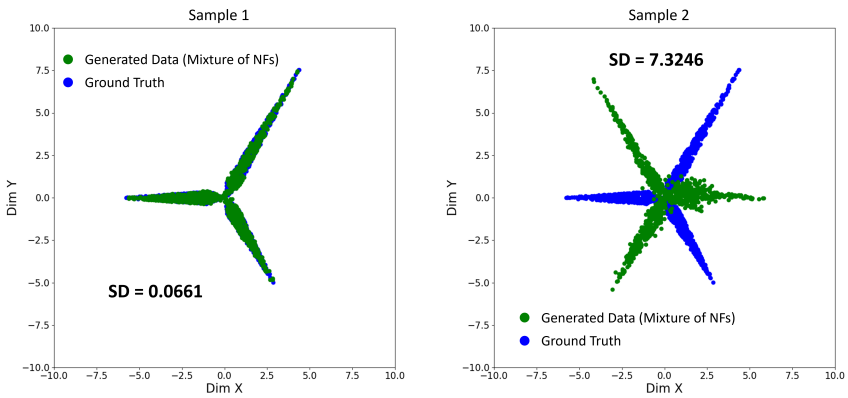


Figure 4.19: Instance of correct sample generation (left) and incorrect sample generation (right) for the condition specifying 3 arms and 60 degrees rotation

Fig. [4.19] shows an instance where two samples are drawn for the condition specifying 3 arms and a 60-degree rotation. Sample 1 correctly corresponds to the given condition, whereas Sample 2 does not. This discrepancy is also reflected in the Sinkhorn distances of these samples.

### 4.4.1 Analysis of the learned latent distributions

This inconsistency in generation is further examined by generating 125 samples for each condition. The posterior and prior samples for each of these conditions are recorded, and Principal Component Analysis (PCA) is performed on these samples. Additionally, the probability of generating a correct sample for each given condition is calculated.

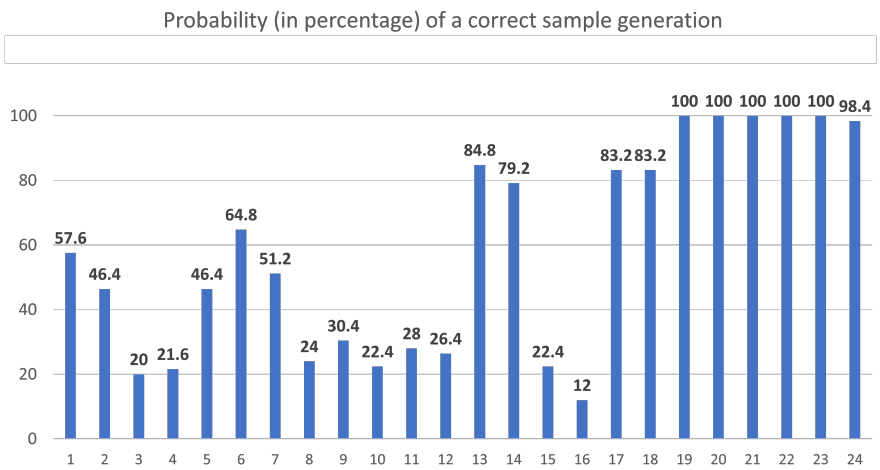


Figure 4.20: Probability of generating the correct sample for each of the 24 conditions.

Fig. [4.20] shows the probability of generating the correct sample for all 24 conditions. Condition 1, corresponding to a 2-armed star with 0 degrees of rotation, has a success rate of 57.4%, whereas conditions 19 to 23, which correspond to 5-armed stars, show a success rate of 100%. A PCA visualization of these samples (Fig. [4.21]) indicates that the prior samples for conditions 1 and 16 are spread out, whereas for condition 19, which shows a 100% success rate, the prior samples cluster at almost the same coordinates in the principal component space.

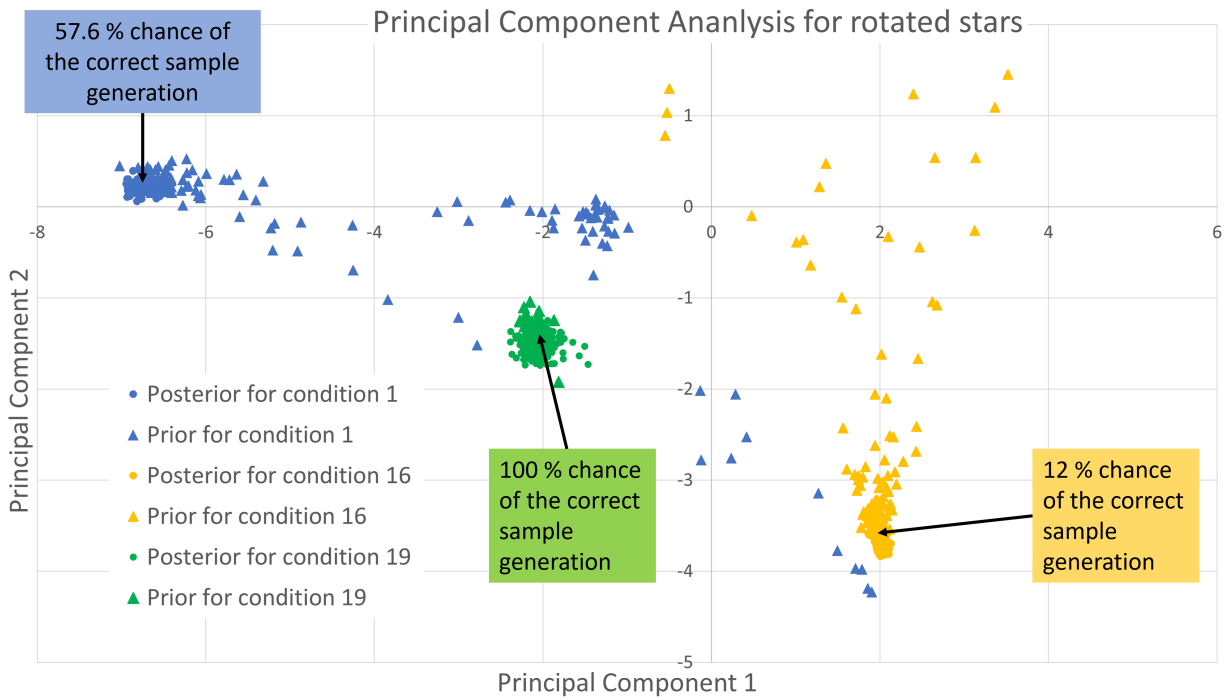


Figure 4.21: PCA visualization of prior samples for selected conditions.

It is also found that the GNLL loss term is a good indicator of clustering. Fig. [4.22] shows a negative correlation between GNLL and the probability of correct sample generation.

This indicates that the prior distribution is not formulated well enough for the model to distinguish various conditions from each other.

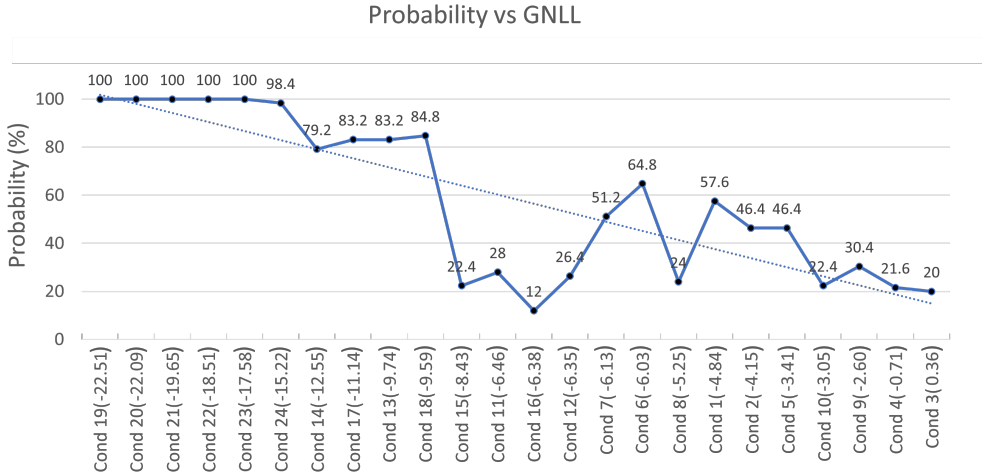


Figure 4.22: Negative correlation between GNLL loss and probability of correct sample generation.

The hypothesis of prior generation being the root cause of the problem is confirmed by carrying out the generation process without prior generation and replacing the prior sample with a posterior sample. Fig. [4.23] shows a few samples that are generated in the autoencoding setup and similar results are observed for all the conditions.

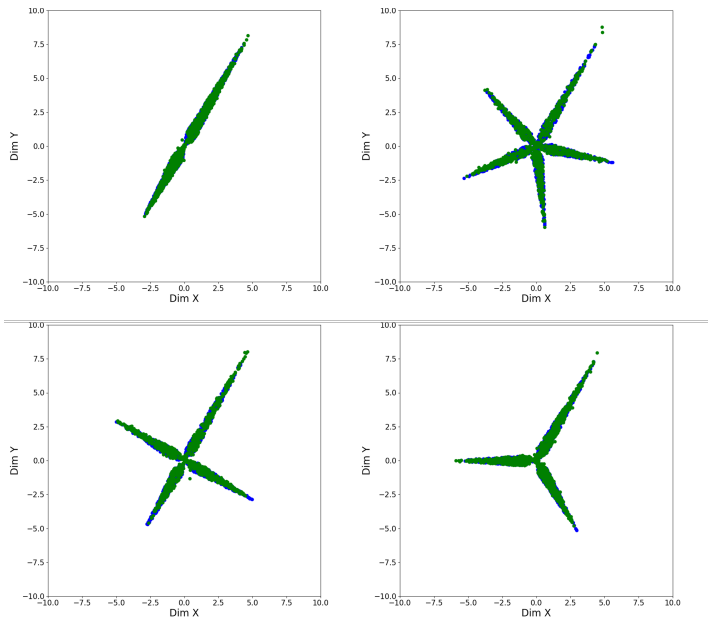


Figure 4.23: Examples of ground truth and the generated data perfectly matching in the autoencoder configuration of the model.

In conclusion, the autoencoding experiment, and the PCA visualizations confirm that the root cause of non-compliance of the model for conditional generation is the prior learning. The decoder, which is responsible for the reconstruction, and the encoder that maps the input point clouds to the posterior space are functional as per design.

# 5 Model Reduction and Training Protocols

Mixture of NFs is designed to learn the distribution of complex-shaped point clouds by specializing in different sub-regions. As discussed in section 3.6, this is achieved through the use of multiple blocks of normalizing flows. Consequently, this complexity results in a significantly high number of trainable parameters necessitating a strong model reduction. Additionally, custom training protocols help reduce the total training time by strategically performing the backpropagation step, the most computationally expensive part of training, at intervals rather than after every training epoch. This chapter explores these techniques in detail.

## 5.1 Model Reduction

The Mixture of Normalizing Flows (NFs) model comprises three main components: the encoder, which maps the input sample to a posterior space; the prior NF, which learns the distribution of these mapped samples; and the decoder, which consists of multiple conditional NFs. Each of these components includes neural networks that require training. Among these three components, the decoder accounts for more than 80% of the total parameters. To balance model complexity with computational efficiency, the feature size and the number of RealNVP coupling blocks in the decoder are optimized. This optimization is guided by heuristics used in the official implementation of the Mixture of NFs. The model reduction is executed in three modes.

1. Depth First: In this mode, the number of coupling blocks is reduced according to the formula $r_d = \frac{1}{m} \times n_d$, where $m$ is the number of decoders and $n_d$ is the initial number of coupling blocks. The reduced number of features $r_f$ is selected such that the number of trainable parameters in each decoder is reduced to $\frac{1}{m}$ of the initial number of parameters.
2. Feature First: In this mode, the number of features is initially reduced until the total number of parameters in the reduced model is $\frac{1}{m}$ times the original model. If the

reduced number of features $r_f$ reaches 4, further reduction of features is stopped, and the number of coupling blocks is reduced instead until the criterion is satisfied.

3. **Depth and Feature**: This mode balances the reduction between the number of coupling blocks and the features. It limits the number of coupling blocks to $\frac{1}{\sqrt{n_d}}$ of the initially defined number of coupling blocks and then uses this reduced number to calculate the reduced features $r_d$.

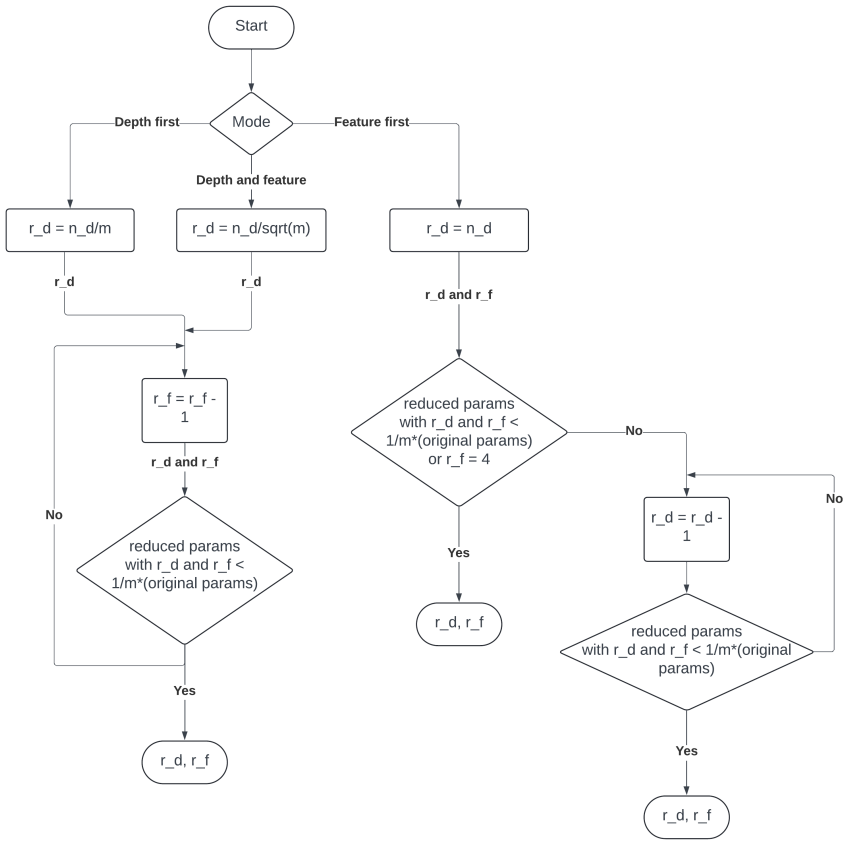


**Figure 5.1:** Flowchart of the model reduction algorithm based on number of flows and features

**Fig. [5.1]** presents a flowchart illustrating these three modes of model reduction. This algorithm specifically addresses the parameters within the decoder and is constrained by the choice of $n_d$. The model size is further optimized by evaluating the quality of the generated point cloud for various values of $n_d$.

**Table 5.1:** Comparison of Model Reduction Modes

| Reduction mode | No. of parameters | Sinkhorn Distance |
|---|---|---|
| Depth first | 2.09 Million | 0.8484 |
| Feature first | 6.25 Million | 0.3609 |
| Depth and Feature | 3.48 Million | 0.3932 |

All reduction modes are tested on a toy dataset with 6 dimensions, with additional features added in each dimension, and the results are summarized in **Tab. [5.1]**. The "Depth and Feature" mode offers the optimal balance between the number of parameters and

reconstruction quality. **Fig. [5.2]** displays the projections of the first two dimensions for the different model reduction modes.

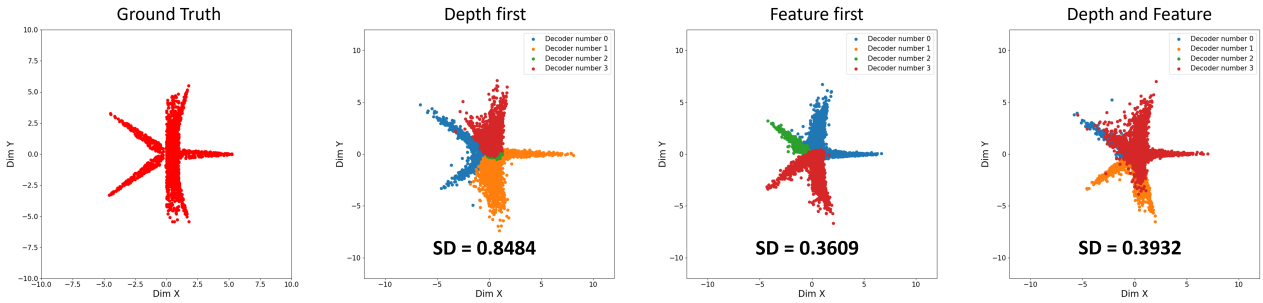


**Figure 5.2:** Projections of the first two dimensions for different model reduction modes

Additionally, the model's performance is tested for various choices of $n_d$. Given the goal of this study to reduce the model size, $n_d$ is decreased, and the "Depth and Feature" mode is selected. The results are presented in **Tab. [5.2]**.

**Table 5.2:** Performance for Different Values of $n_d$

| $n_d$ | $r_d$ | No. of parameters | Sinkhorn Distance |
|---|---|---|---|
| 21 | 11 | 3.48 Million | 0.3932 |
| 14 | 7 | 2.37 Million | 0.4651 |
| 7 | 4 | 1.54 Million | 1.2389 |
| 3 | 2 | 0.99 Million | 1.3350 |

The corresponding results, shown as projections of the first two dimensions, are illustrated in **Fig. [5.3]**. With 2.37 million parameters, an $n_d$ value of 14, combined with the "Depth and Feature" mode, achieves the optimal balance between the number of parameters and reconstruction quality.

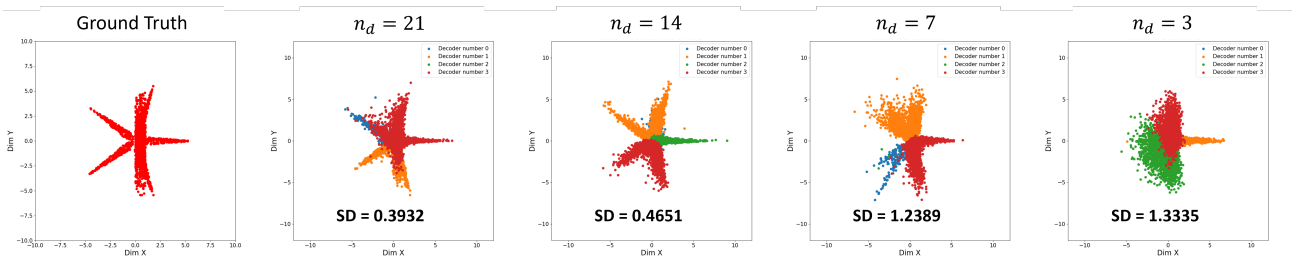


**Figure 5.3:** Projections of the first two dimensions for different values of $n_d$

In conclusion, among the three modes of model reduction, the 'Depth and feature' mode is chosen as it provides the optimal trade-off between reconstruction quality and the number of parameters. Further, the number of coupling blocks that the model is initialized with is reduced to 14 from 21, this effectively decreases the number of trainable parameters by 43%. This makes the Mixture of NFs model more feasible for surrogate modeling of processes in plasma physics.

## 5.2 Training Protocols

In addition to model reduction, another effective technique to make the Mixture of NFs-based surrogate models less computationally expensive is the design of custom training protocols that selectively update the components of the machine learning model with the gradients. In other words, backpropagation is performed selectively and not for every epoch. This section presents an initial study that examines the training time of two customized training protocols and compares it with the baseline training procedure.

The input dataset for this study comprises 3D point clouds in the shape of stars with different angular and shape configurations. The input data consists of 24 conditions, where each condition corresponds to a point of particular shape and rotation. Since the distribution of the number of trainable parameters is skewed towards the decoder, only the decoder is subjected to selective updating of the parameters. The other parts of the model are updated for every epoch. The details of these protocols are as follows:

- **Baseline Protocol:** Backpropagation is done for every epoch. Consequently, this training procedure takes the longest time.
- **Protocol 1:** In this mode, the parameters of the decoder are updated once every 5 epochs. This represents a very strong form of training time reduction and therefore achieves the least amount of training time as the decoder is updated only 20% of the time at regular intervals.
- **Protocol 2:** This mode corresponds to a gradual decrease in the number of backpropagations. For the first 200 epochs, the decoder is updated with the gradients for all epochs. For the next 200 epochs, the decoder is updated for alternate epochs. In the following 200 epochs, the decoder parameters are updated every 2 epochs, and so on, until the decoder parameters are updated once every 5 epochs. This protocol takes a gradual approach to the reduction of backpropagation cycles and aims to achieve a trade-off between training time and the maximum frequency of backpropagation.

**Table 5.3:** Training time for different protocols

| Mode | Training time |
|---|---|
| Baseline Protocol | 1 hour 58 min |
| Protocol 1 | 1 hour 27 min |
| Protocol 2 | 1 hour 40 min |

The training time for 1000 epochs is tabulated for each protocol in **Tab. [5.3]** with no change in the reconstruction quality as seen in **Fig. [5.4]**.

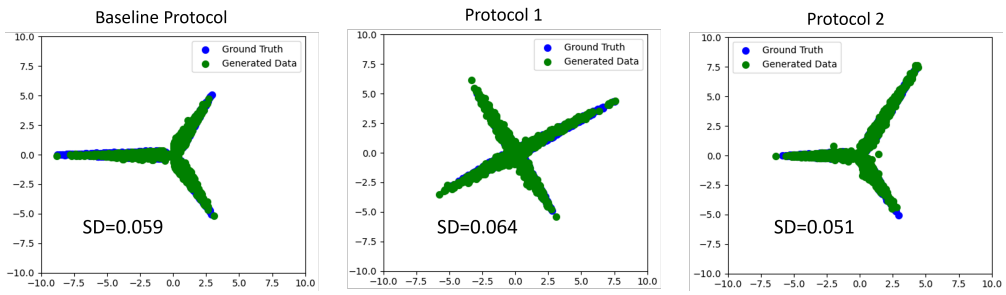


Figure 5.4: Projections of the first two dimensions for different training protocols with corresponding Sinkhorn distance values

# 6 Application of Mixture of NFs to Plasma Physics Process

This chapter explores the application of Mixture of NFs to a plasma physics process, specifically in the context of a free electron laser [Pas]. The training data comprises point cloud configurations of the electron bunch at various stages of the FEL. Simulation parameters, along with the diagnostic stage of the FEL, are used to define the input conditions. The model is trained on this simulation data with scalar and one-hot vector forms of conditioning. The corresponding point clouds are generated and the results are then analyzed both qualitatively and quantitatively to evaluate the reconstruction quality and the model's adherence to the specified conditions. Principal Component Analysis (PCA) is performed on the model's outputs to further explain the conditionality.

## 6.1 Free-Electron Laser

An FEL is an advanced light source known for emitting exceptionally bright and brief radiation pulses. While FELs operate similarly to traditional lasers, they differ in their gain medium, using relativistic electrons instead of atomic or molecular excitations. In an FEL, a bunch of electrons travels through a magnetic structure called an undulator or wiggler, producing radiation. The interaction between this radiation and the electron bunch leads to coherent emission from the electrons, resulting in an exponential increase in overall radiation intensity.

To minimize the size of the facility, laser-plasma acceleration (LPA) is employed rather than conventional linear accelerators like the ones used in SLAC's accelerator. LPA is a highly non-linear process that defines the radiation characteristics in the undulator. The accelerated beam is directed through optics to the undulator, where it emits radiation.

A significant diagnostic challenge in this setup is that the beam can only be observed once in the transport line, as passing it through the imager is destructive to the beam. Therefore, the beam transport of the accelerated beam from LPA must be simulated along with its radiation intensity in the undulator. The simulation needs to match collected diagnostic data to reconstruct a complete experiment in detail. A surrogate model can expedite this

simulation, helping to find suitable parameters more quickly and thus accelerating the entire experimental process.

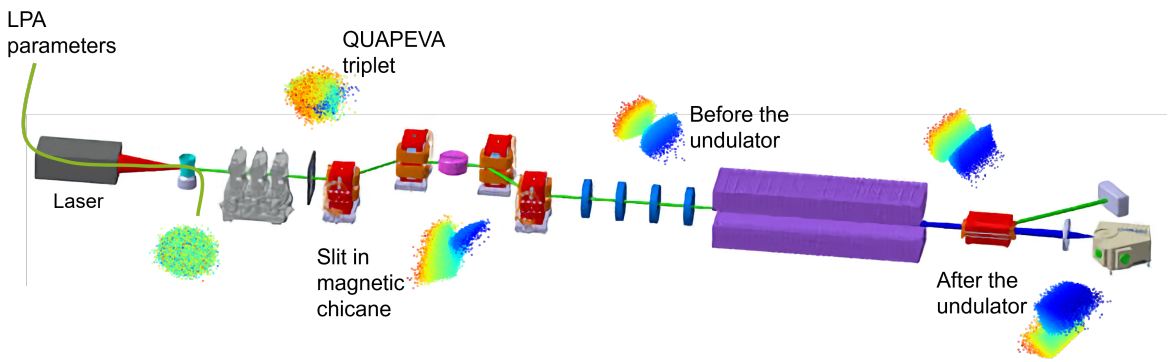


Figure 6.1: Experimental setup of a Free Electron Laser along with electron bunch configurations at each stage of the beamline

Fig. [6.1] depicts the simulation results of an FEL beam transport where the point cloud configurations correspond to various diagnostic stages of the laser. Each of these point clouds is six-dimensional. The first three dimensions give the spatial coordinates, the next two correspond to the per particle divergence in $X$ and $Y$ directions, and the final dimension gives the energy of the particles. This training data set is formulated with 1024 conditions that correspond to different simulation parameters and stages of an FEL beamline. The mixture of NFs model is trained on this data set and the results are analyzed.

## 6.2 Generated data: Scalar Conditioning

The generated data which is six-dimensional with scalar conditioning is analyzed using various visualization techniques. The reconstruction quality and the geometric accuracy are checked by measuring the Sinkhorn distance and by qualitatively examining the 2D projections of the generated point cloud. The learned distribution is examined by comparing the histograms of each dimension and the conditionality is checked by performing PCA on the prior samples and posterior samples.

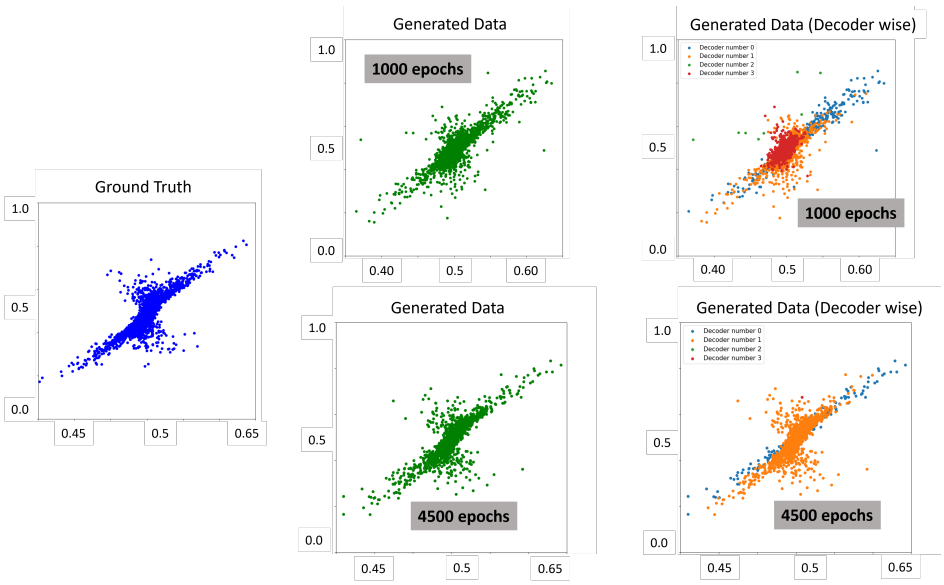


Figure 6.2: Z-coordinate vs Per particle divergence in x-direction of the ground truth (left) corresponding to Imager 6, generated data by mixture of NFs model (center), and the decoder wise points distribution of the generated data (right) given by different colors

The two-dimensional projections of the generated data are plotted as seen in Fig. [6.2]. This figure demonstrates that for the initial stages of training, different decoders reconstruct corresponding sub-regions of the point cloud but with further training, only a few decoders are responsible for the majority of the reconstructed point cloud without compromising the reconstruction quality. This suggests that the number of decoders can be reduced which also reduces the model complexity depending upon the application.

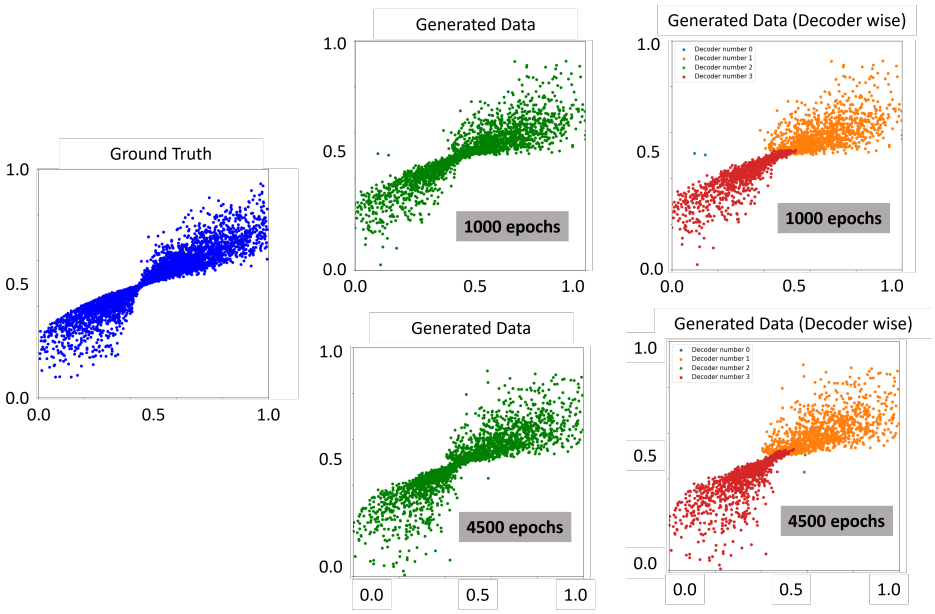


Figure 6.3: X-coordinate vs. Y-coordinate of the ground truth (left) corresponding to Imager 5, generated data by mixture of NFs model (center), and the decoder wise points distribution of the generated data (right) given by different colors

Fig. [6.3] shows that the complexity in the point cloud is resolved by decoders learning only the sub-regions of the point cloud as per design. This shows that one of the goals

of employing the Mixture of NFs model to resolve complex-shaped point clouds into sub-regions is accomplished.
The distribution of points in each dimension is studied by plotting the histograms of the generated data and the simulation data which is used for training. One of such generated data is analyzed using this technique and the results are shown in Fig. [6.4].

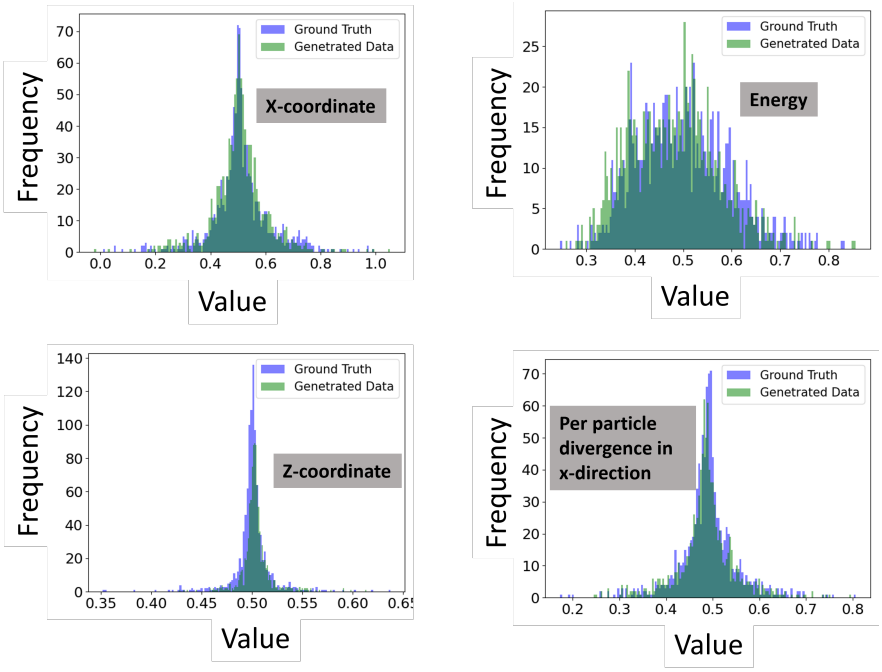


Figure 6.4: Histograms of generated data of Imager 5 in comparison with the learned distribution by the Mixture of NFs model for FEL data conditioned over scalar values

It is also observed that the generated data, when correctly corresponding to a given condition, closely resembles the distribution of the simulation data across dimensions as seen in Fig. [6.4].
However, the model exhibits inconsistency with generating the correct point cloud configuration for a specific condition as shown in Fig. [6.5]. Sample 1 correctly corresponds to the ground truth and sample 2 has a configuration of a point cloud that is given by a different condition.

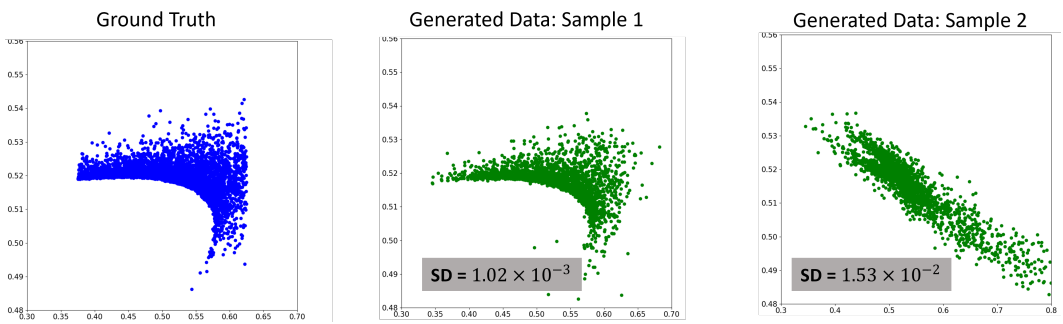


Figure 6.5: Example of inconsistent sample generation

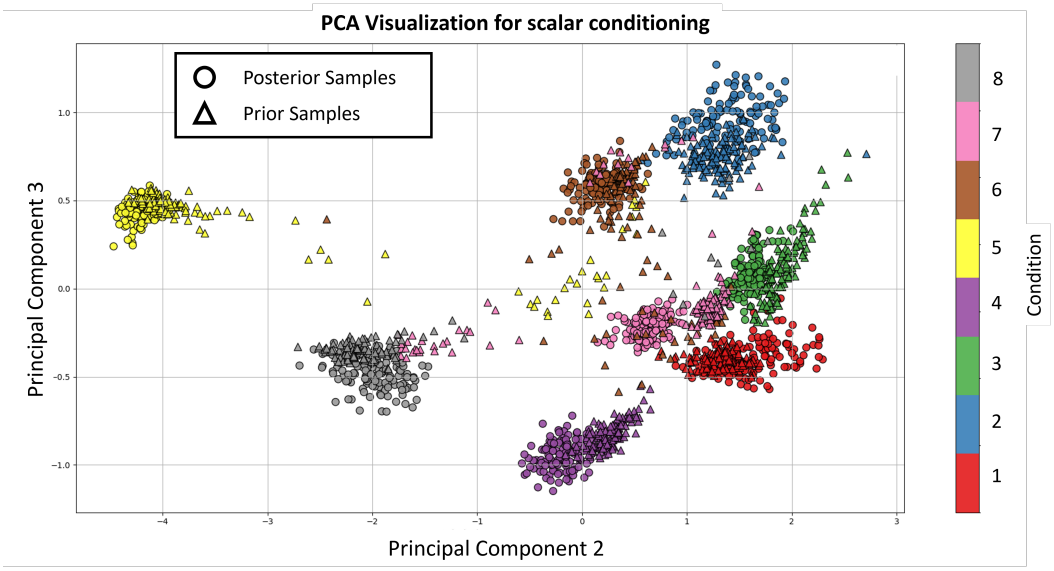


Figure 6.6: PCA visualization of prior samples for FEL data for scalar conditioning at 4500 epochs

As it was analyzed in the previous experiments, the probability of generating the right sample is calculated for each condition as seen in Fig. [6.7]. This is indicative of the prior distribution that is learned for each condition. The PCA visualization is shown in Fig. [6.6] reveals that with a success rate of 100% along with other conditions like condition 2, condition 3, etc. The model generates samples that correspond to these conditions better than the other. This is a consequence of the prior distribution being relatively more similar to the corresponding posterior as shown in Fig. [6.6]. On the other hand, the samples corresponding to condition 7 are the least reliable as the prior distribution is scattered and doesn't comply with the respective posterior distribution.

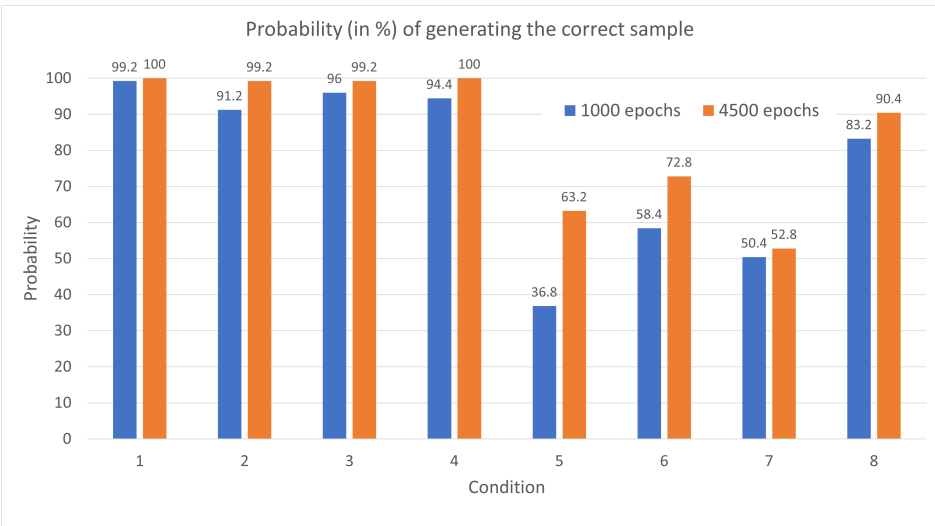


Figure 6.7: Probability of generating the correct sample for each condition (Imager)

## 6.3 Generated data: One-hot Vector Conditioning

In this mode of training, the model is conditioned on one-hot vectors that represent different stages of the FEL beamline. The scalar conditioning exhibits inconsistency in terms of recognizing these conditions and often takes longer training time to learn these conditions. The performance of the one-hot vector conditioning is first assessed in terms of the reconstruction quality.

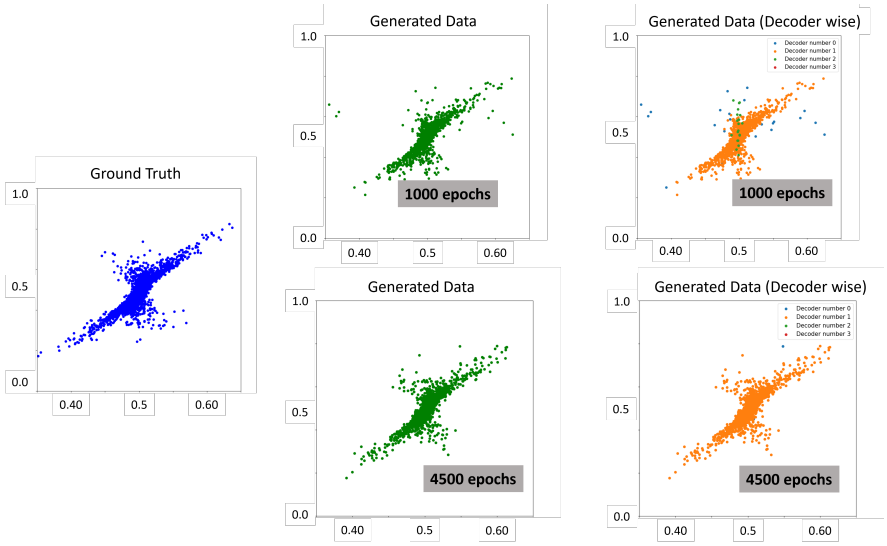


Figure 6.8: Z-coordinate vs Per particle divergence in x-direction of the ground truth (left) corresponding to Imager 6, generated data by mixture of NFs model (center), and the decoder wise points distribution of the generated data (right)

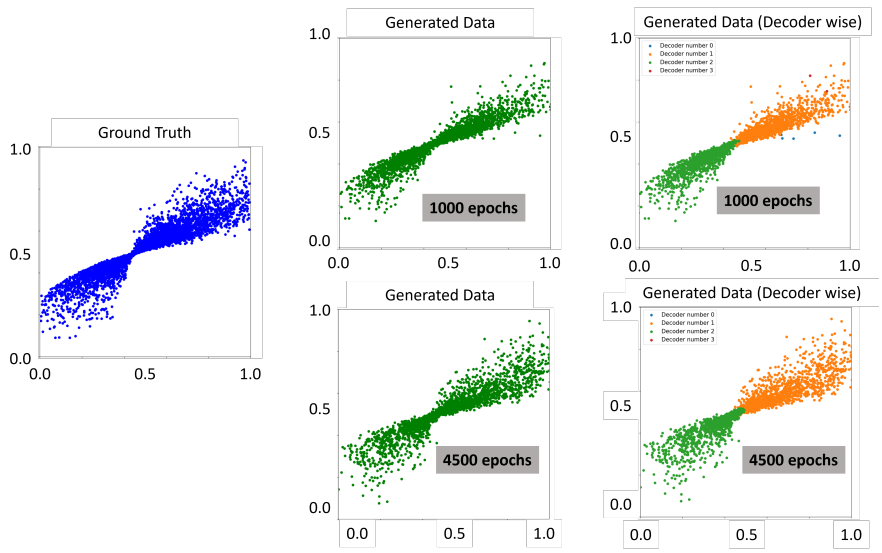


Figure 6.9: X-coordinate vs Y-coordinate of the ground truth (left) corresponding to Imager 5, generated data by mixture of NFs model (center), and the decoder wise points distribution of the generated data (right) given by different colors

The model performs similarly to scalar conditioning in terms of reconstruction quality. The early stages of the training show that the number of points is distributed between the

decoders. As the training progresses, the number of decoders needed to capture the entire point cloud for a specific condition decreases. The average Sinkhorn distances and the associated standard deviations are tabulated in Tab. [6.1] and Tab. [6.2]. Fig. [6.9] shows that the one-hot conditioning also enables the model to specialize in sub-regions to generate a point cloud that complies with the input.

Table 6.1: Average Sinkhorn distance values and their standard deviations for **scalar conditioning** at various training durations across different conditions.

| | Mean SD (1000 epochs) | Std Dev. (1000 epochs) | Mean SD (4500 epochs) | Std Dev. (4500 epochs) |
|---|---|---|---|---|
| Condition 1 | $1.53 \times 10^{-3}$ | $6.35 \times 10^{-4}$ | $1.10 \times 10^{-3}$ | $2.29 \times 10^{-4}$ |
| Condition 2 | $2.38 \times 10^{-3}$ | $3.82 \times 10^{-3}$ | $1.39 \times 10^{-3}$ | $1.28 \times 10^{-3}$ |
| Condition 3 | $2.05 \times 10^{-3}$ | $1.47 \times 10^{-3}$ | $1.61 \times 10^{-3}$ | $6.51 \times 10^{-4}$ |
| Condition 4 | $2.58 \times 10^{-3}$ | $2.78 \times 10^{-3}$ | $1.35 \times 10^{-3}$ | $4.74 \times 10^{-4}$ |
| Condition 5 | $2.22 \times 10^{-2}$ | $1.91 \times 10^{-2}$ | $1.55 \times 10^{-2}$ | $1.84 \times 10^{-2}$ |
| Condition 6 | $6.02 \times 10^{-3}$ | $7.56 \times 10^{-3}$ | $4.80 \times 10^{-3}$ | $9.28 \times 10^{-3}$ |
| Condition 7 | $1.02 \times 10^{-2}$ | $1.07 \times 10^{-2}$ | $1.07 \times 10^{-2}$ | $1.06 \times 10^{-2}$ |
| Condition 8 | $3.86 \times 10^{-3}$ | $4.43 \times 10^{-3}$ | $2.86 \times 10^{-3}$ | $3.82 \times 10^{-3}$ |

Table 6.2: Average Sinkhorn distance values and their standard deviations for **one-hot vector conditioning** at various training durations across different conditions.

| | Mean SD (1000 epochs) | Std Dev. (1000 epochs) | Mean SD (4500 epochs) | Std Dev. (4500 epochs) |
|---|---|---|---|---|
| Condition 1 | $1.38 \times 10^{-3}$ | $2.63 \times 10^{-4}$ | $1.70 \times 10^{-3}$ | $7.47 \times 10^{-4}$ |
| Condition 2 | $1.13 \times 10^{-3}$ | $4.79 \times 10^{-4}$ | $1.68 \times 10^{-3}$ | $9.65 \times 10^{-4}$ |
| Condition 3 | $1.97 \times 10^{-3}$ | $4.34 \times 10^{-4}$ | $1.99 \times 10^{-3}$ | $6.21 \times 10^{-4}$ |
| Condition 4 | $1.37 \times 10^{-3}$ | $3.39 \times 10^{-4}$ | $1.64 \times 10^{-3}$ | $8.83 \times 10^{-4}$ |
| Condition 5 | $8.00 \times 10^{-3}$ | $3.40 \times 10^{-3}$ | $4.52 \times 10^{-3}$ | $1.39 \times 10^{-3}$ |
| Condition 6 | $4.71 \times 10^{-3}$ | $2.01 \times 10^{-3}$ | $1.52 \times 10^{-3}$ | $6.36 \times 10^{-4}$ |
| Condition 7 | $1.69 \times 10^{-3}$ | $3.06 \times 10^{-4}$ | $1.72 \times 10^{-3}$ | $7.51 \times 10^{-4}$ |
| Condition 8 | $2.83 \times 10^{-3}$ | $1.54 \times 10^{-3}$ | $2.12 \times 10^{-3}$ | $1.02 \times 10^{-3}$ |

In comparison with the scalar conditioning, a significant reduction in the average Sinkhorn distance and the standard deviation is observed over a sample size of 125, especially for conditions 5, 6, and 7. To check the performance of the model in terms of how well it obeys conditioning, principal component analysis of the posterior and prior samples is performed and visualized in Fig. [6.10]

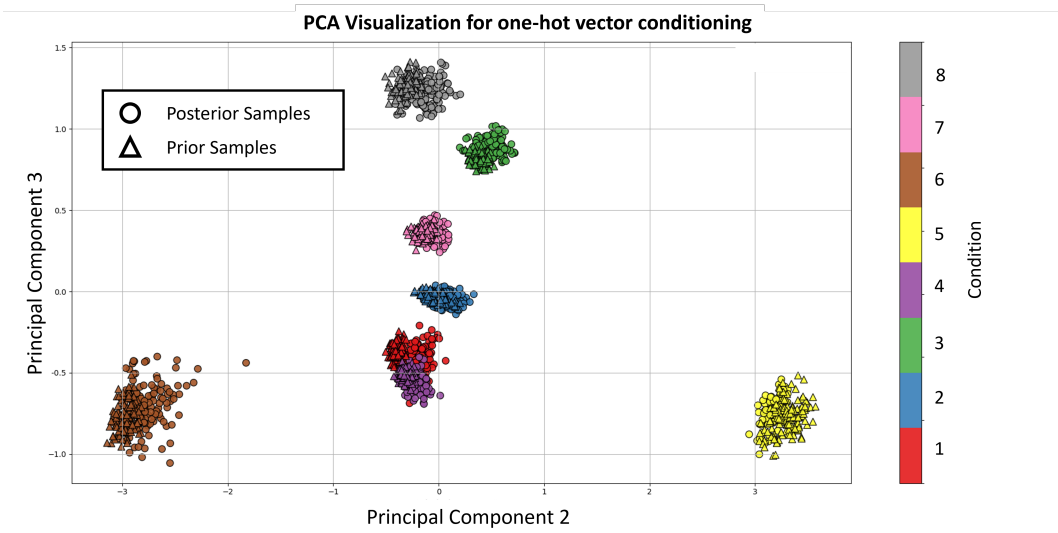


Figure 6.10: PCA visualization of prior samples for FEL data for one-hot vector conditioning at 4500 epochs

The visualization shown in Fig. [6.10] shows that the prior distribution is learned much more accurately in comparison with scalar conditioning. This is also seen in the average Sinkhorn distance recorded for all the conditions. Examination of the generated point clouds also suggests that the condition is obeyed for all the 125 samples suggesting a higher reliability in comparison with scalar conditioning.

# 7 Conclusion

In the concluding chapter, the evaluation of experiments conducted on the generative model of a mixture of normalizing flows (NFs) is presented. The model's performance in reconstructing point clouds is analyzed, along with assessments of resource consumption and recommended techniques for optimizing computational resources. The application of the model to an FEL is also explored, with a comprehensive evaluation of the results. Additionally, the chapter discusses the advantages and disadvantages of this model in detail.

## 7.1 Discussion

The architecture of the Mixture of normalizing flows is used as a surrogate model for processes in plasma physics. This generative model for point clouds learns the distributions of point clouds by specializing in different sub-regions, utilizing individual normalizing flows for these sub-regions. The model was adapted to incorporate conditional information and generate corresponding point cloud distributions. This conditioning is done in two forms: scalar conditioning and one-hot vector conditioning.

Through experimentation on test and FEL simulation data, it is evident that the adapted model can generate complex point clouds by using normalizing flows to reconstruct different parts of the target point cloud distribution. Consequently, it generally produces a better distribution of point clouds compared to a single NF model the number of points that are generated by each normalizing flow block is significantly reduced. This equips the Mixture of NFs-based models to generate point clouds that are more complex in geometrical aspects in comparison to their single-NF counterparts. The experiments also revealed that the model is sensitive to density variations and can successfully learn and reproduce these variations. The model generally respects conditionality by estimating different prior distributions for each condition. Scalar conditioning is useful for compactly encoding class-specific information, while one-hot vector conditioning helps the model adhere more strongly to the conditioning. Both forms of conditioning can be used for the Mixture of NFs. While the reconstruction performance is similar for both forms of conditioning, one-hot vector conditioning is observed

to be better suited for estimating the prior distributions for each condition. Additionally, this estimation is achieved in a much shorter training time compared to scalar conditioning.

However, as the model is more sophisticated, with different components responsible for learning the target distribution and generating it part-wise, this sophistication comes at the cost of computational resources. The model requires much longer training times compared to its single NF counterparts, resulting in much higher computational costs. Experimentation also indicates that the number of decoders plays a crucial role in determining both the accuracy of the generated point cloud and the computational costs but the number of decoders is user-defined. This further demands optimizing the number of decoders as this number is application-specific. In terms of conditioning, it was observed that scalar conditioning requires longer training time in comparison with one-hot vector conditioning to learn the condition-specific prior distribution. Thus, for relatively shorter training periods, the scalar conditioning produces prior distributions that do not comply with the given condition. Single NFs trained on scalar conditioning yield point clouds of intermediate configuration between two defined condition-generated point-cloud pairs. This behavior is not observed in the mixture of normalizing flows, as the decoder is conditioned on samples from the prior distribution, which are in turn conditioned on the scalar values. Finally, since the division of decoders in generating the point cloud is unsupervised, there is no control over how the generated point cloud is split between the decoders.

## 7.2 Summary

Simulations in plasma physics are often computationally intensive and require numerous trials to bridge the gap between simulation results and experimental observations. To facilitate better in-situ analysis, data-driven surrogate models have been proposed as a viable solution. This approach is realized in the form of generative models for point clouds, which take simulation parameters as conditions and output the corresponding point cloud distributions.

The architecture of a mixture of normalizing flows has been identified as particularly suitable for this purpose, as it is specifically designed to learn point cloud distributions with complex shapes. The model has been tested on simple two-dimensional and three-dimensional data, demonstrating its capability to learn the distribution by splitting the target distribution into parts at areas of discontinuities. Furthermore, the was adapted to incorporate conditionality in two forms: scalar conditioning and one-hot vector conditioning by replacing image conditioning with class-specific conditioning. The resulting model was compared in terms of reconstruction quality with single NF models and was found that the Mixture of NFs performs generally better on these test distributions. The model's ability to accommodate input data of higher dimensions was also compared with the performance of single NF models and similar results were observed. The model could also specialize in different sub-regions in the higher dimensions as well. Both these models were also tested on density variations and it was observed that they preserve the variations in the densities. The mixture of NFs model was subjected to an increased number of conditions and it was found that for scalar conditioning, the model failed to reliably produce point clouds in compliance with the given condition. To further investigate this, the learned prior distribution was visualized with the

help of principal component analysis. It was observed that the probability of generating the correct sample for a condition is negatively correlated with the GNLL component of the loss function. In the experiments so far, it was observed that the Mixture of NFs model, due to its architecture, has a high number of trainable parameters making it computationally inefficient. The heuristic-based model reduction algorithm that was already a part of the official implementation of the model was studied and together with further optimization of the number of coupling blocks in the decoder, a strong reduction in the number of trainable parameters was achieved. To further reduce the training time, custom training protocols were implemented and were tested on toy data sets. It was observed that these protocols were effective in bringing down the training time. Finally, the reduced model was applied to the simulation data of an FEL with scalar and one-hot vector conditioning. The results were analyzed for different training lengths and types of conditioning. It was observed that the scalar conditioning does not reliably respect the given condition. Although the probability of generating the correct sample improves with longer training periods, it is still outperformed by one-hot vector conditioning. Analysis of the learned prior distribution revealed that one-hot vector conditioning aids the model in consistently generating point clouds in compliance with the given condition. This was further confirmed by the mean and standard deviation values of the Sinkhorn distances measured for the generated data. As a result, a sophisticated surrogate model was successfully formulated. This model can be employed to conduct faster in-situ analysis for experiments in plasma physics, offering a promising solution to the challenge of computationally intensive simulations.

## 7.3 Outlook

The adapted surrogate model demonstrates effective reconstruction of point clouds based on given conditions, yet there remains considerable room for improvement. Currently, the distribution of trainable parameters is heavily skewed towards the decoder, and determining the optimal number of decoders requires empirical methods. Future research could focus on developing techniques to predict the appropriate number of decoders, thereby optimizing the allocation of trainable parameters.

The custom training protocols introduced in this work are based on heuristic approaches. Implementing these protocols in practical surrogate models for plasma physics processes could significantly reduce training time. Further investigations into these training protocols and their impact could enhance model performance, leading to more efficient and accurate surrogate models for plasma physics.

Additionally, architectural modifications that facilitate the interpolation of point cloud configurations between two given conditions could be introduced. Successfully formulating such a model would enable the description of all intermediate states between two point cloud configurations.

The Mixture of NFs model is particularly sensitive to the choice of the number of decoders, which is currently a user-defined parameter. Developing a technique to determine this number based on the geometry of the point cloud would further enhance the model's performance.

# Bibliography


[MP43] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[Che74] F. F. Chen. *Introduction to plasma physics*. New York: Plenum Press, 1974.

[Hay94] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[Roj96] Raúl Rojas. "Threshold Logic". In: *Neural Networks: A Systematic Introduction*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 29–53. ISBN: 978-3-642-61068-4. DOI: 10.1007/978-3-642-61068-4_2. URL: https://doi.org/10.1007/978-3-642-61068-4_2.

[HS06] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786 (2006), pp. 504–507. DOI: 10.1126/science.1127647. URL: https://www.science.org/doi/abs/10.1126/science.1127647.

[Mad+11] Kamesh Madduri et al. "Gyrokinetic toroidal simulations on leading multi- and manycore HPC systems". In: Nov. 2011, p. 23. DOI: 10.1145/2063384.2063415.

[Cut13] Marco Cuturi. *Sinkhorn Distances: Lightspeed Computation of Optimal Transportation Distances*. 2013. arXiv: 1306.0895 [stat.ML].

[Goo+14] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[He+15] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[Car+16] G. Carneiro et al. "Deep Learning and Data Labeling for Medical Applications". In: *Lecture Notes in Computer Science*. 2016. URL: https://api.semanticscholar.org/CorpusID:26585209.

[RM16] Danilo Jimenez Rezende and Shakir Mohamed. *Variational Inference with Normalizing Flows*. 2016. arXiv: 1505.05770 [stat.ML]. URL: https://arxiv.org/abs/1505.05770.

[Sal+16] Tim Salimans et al. *Improved Techniques for Training GANs*. 2016. arXiv: 1606.03498 [cs.LG].

[DSB17] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. *Density estimation using Real NVP*. 2017. arXiv: 1605.08803 [cs.LG].

[Qi+17] Charles R. Qi et al. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2017. arXiv: 1612.00593 [cs.CV]. URL: https://arxiv.org/abs/1612.00593.

[Ach+18] Panos Achlioptas et al. *Learning Representations and Generative Models for 3D Point Clouds*. 2018. arXiv: 1707.02392 [cs.CV].

[Ard+18] Lynton Ardizzone et al. *Framework for Easily Invertible Architectures (FrEIA)*. 2018. URL: https://github.com/vislearn/FrEIA.

[GWM18] Matheus Gadelha, Rui Wang, and Subhransu Maji. *Multiresolution Tree Networks for 3D Point Cloud Processing*. 2018. arXiv: 1807.03520 [cs.CV].

[Li+18] Chun-Liang Li et al. *Point Cloud GAN*. 2018. arXiv: 1810.05795 [cs.LG].

[TW18] Jakub M. Tomczak and Max Welling. *VAE with a VampPrior*. 2018. arXiv: 1705.07120 [cs.LG].

[Ard+19] Lynton Ardizzone et al. *Guided Image Generation with Conditional Invertible Neural Networks*. 2019. arXiv: 1907.02392 [cs.CV].

[Yan+19] Guandao Yang et al. *PointFlow: 3D Point Cloud Generation with Continuous Normalizing Flows*. 2019. arXiv: 1906.12320 [cs.CV].

[Zam+19] Maciej Zamorski et al. *Adversarial Autoencoders for Compact Representations of 3D Point Clouds*. 2019. arXiv: 1811.07605 [cs.LG].

[Cra+20] Miles Cranmer et al. *Discovering Symbolic Models from Deep Learning with Inductive Biases*. 2020. arXiv: 2006.11287 [cs.LG].

[Din+20] Laurent Dinh et al. *A RAD approach to deep mixture models*. 2020. arXiv: 1903.07714 [cs.LG].

[Sti+20] Patrick Stiller et al. *Large-scale Neural Solvers for Partial Differential Equations*. 2020. arXiv: 2009.03730 [cs.LG].

[Sty+20] Michał Stypułkowski et al. *Representing Point Clouds with Generative Conditional Invertible Flow Networks*. 2020. arXiv: 2010.11087.

[Cor+21] Rob Cornish et al. *Relaxing Bijectivity Constraints with Continuously Indexed Normalising Flows*. 2021. arXiv: 1909.13833 [stat.ML].

[Don+21] G. Dong et al. "Deep learning based surrogate models for first-principles global simulations of fusion plasmas". In: *Nuclear Fusion* 61.12 (Nov. 2021), p. 126061. ISSN: 1741-4326. DOI: 10.1088/1741-4326/ac32f1. URL: http://dx.doi.org/10.1088/1741-4326/ac32f1.

[Pat+21] Bhavesh G Patel et al. "A 3D magnetohydrodynamic simulation of the propagation of a plasma plume transverse to applied magnetic field". In: *Plasma Physics and Controlled Fusion* 63.11 (Oct. 2021), p. 115020. ISSN: 1361-6587. DOI: 10.1088/1361-6587/ac2617. URL: http://dx.doi.org/10.1088/1361-6587/ac2617.

[Pos+21] Janis Postels et al. *Go with the Flows: Mixtures of Normalizing Flows for Point Cloud Generation and Reconstruction*. 2021. arXiv: 2106.03135 [cs.CV].

[KW22] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2022. arXiv: 1312.6114 [stat.ML].

[Wil22] Anna Willmann. *Beam transport setup*. courtesy: https://www.helmholtz.ai/applied-ai/ai-research-labs/kelling-team/. 2022.

[Pas] R. Paschotta. *Free Electron Lasers*. RP Photonics Encyclopedia. Available online at https://www.rp-photonics.com/free_electron_lasers.html. DOI: 10.61835/iqu. URL: https://www.rp-photonics.com/free_electron_lasers.html (visited on 07/10/2024).