Dissertation

submitted to the

Combined Faculty of

Mathematics, Engineering and Natural Sciences

of

Heidelberg University, Germany

for the degree of

Doctor of Natural Sciences

Put forward by

Peter Sorrenson

Born in: Bloomington, Indiana, USA
Date of oral examination: 22 January 2025

Under the supervision of

Prof. Dr. Tilman Plehn
Prof. Dr. Ullrich Köthe

Free-Form Flows: Generative Models for Scientific Applications

Referees: Prof. Dr. Tilman Plehn
Prof. Dr. Fred Hamprecht

Abstract

This thesis explores advanced generative modeling techniques with a focus on scientific applications. It addresses two main areas: free-form flows and machine learning applications in particle physics.

Free-form flows are a novel family of generative models that combine the benefits of normalizing flows—exact maximum likelihood training and fast generation—with unrestricted neural network architectures. This approach overcomes the limitations of traditional normalizing flows, allowing for more flexible and domain-specific models.

The thesis also presents a collection of machine learning applications in particle physics, leveraging models with rich representational spaces. These techniques aim to accelerate the processing of vast data streams from the Large Hadron Collider, potentially uncovering new physics beyond the Standard Model.

By advancing both the theoretical foundations and practical implementations of generative models, this work contributes to their increased adoption and impact across diverse scientific disciplines. Applications range from chemistry to particle physics, demonstrating the broad potential of these methods in modeling complex data distributions.

Diese Dissertation untersucht fortgeschrittene Techniken der generativen Modellierung mit Fokus auf wissenschaftliche Anwendungen. Sie befasst sich mit zwei Hauptbereichen: Free-Form Flows und maschinelle Lernverfahren in der Teilchenphysik.

Free-Form Flows sind eine neuartige Familie generativer Modelle, die die Vorteile von normalisierenden Flows (exaktes Maximum-Likelihood-Training und schnelle Generierung) mit unbeschränkten neuronalen Netzwerkarchitekturen kombinieren. Dieser Ansatz überwindet die Beschränkungen traditioneller normalisierender Flows und ermöglicht flexiblere und domänenspezifische Modelle.

Die Arbeit präsentiert zudem eine Sammlung von Anwendungen des maschinellen Lernens in der Teilchenphysik, die Modelle mit reichhaltigen Repräsentationsräumen nutzen. Diese Techniken zielen darauf ab, die Verarbeitung der enormen Datenströme des Large Hadron Colliders zu beschleunigen und möglicherweise neue Physik jenseits des Standardmodells aufzudecken.

Durch die Weiterentwicklung sowohl der theoretischen Grundlagen als auch der praktischen Implementierungen generativer Modelle trägt diese Arbeit zu ihrer verstärkten Anwendung und Wirkung in verschiedenen wissenschaftlichen Disziplinen bei. Die Anwendungen reichen von der Chemie bis zur Teilchenphysik und demonstrieren das breite Potenzial dieser Methoden bei der Modellierung komplexer Datenverteilungen.

Acknowledgements

I would like to express my gratitude to my supervisors, Prof. Dr. Ullrich Köthe and Prof. Dr. Tilman Plehn, for their guidance and support throughout my doctoral studies, as well as their willingness to allow me to freely explore my interests.

I would like to thank my colleagues and collaborators, particularly Felix Draxler and Barry Dillon. Warm thanks also go to the entire CVL team, the Plehn group, and all the other students I interacted with throughout my PhD, especially Armand Rousselot, Sander Hummerich, Daniel Behrend-Uriarte, Jens Müller, Robert Schmier, Stefan Radev, Lynton Ardizzone, Luigi Favaro, Nathan Huetsch, Sofia Palacios Schweitzer, and Jonas Spinner. To everyone: thank you for all the stimulating discussions and the interesting conversations during lunch breaks.

Most importantly, I thank my wife, Anna, and our baby, Enrico. Without you, I would never have been so motivated to finish! Enrico, it has been a pleasure completing this thesis with you sleeping peacefully on my chest.

Funding Statement

I received funding from the following sources provided by the Deutsche Forschungsgemeinschaft (German Research Foundation):

- Research Training Group GK-1940, *Particle Physics Beyond the Standard Model*.
- Heidelberg STRUCTURES Cluster of Excellence (Germany's Excellence Strategy EXC-2181/1 - 390900948).

Additionally, the following computational resources were utilized to conduct some of the experiments reported in this thesis:

- Support from the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) via grant INST 35/1597-1 FUGG.
- The Center for Information Services and High-Performance Computing (ZIH) at TU Dresden.

Contents

Chapter 1

Introduction

At the Large Hadron Collider, billions of particle collisions generate petabytes of data, yet discovering new physics requires extracting subtle patterns from this vast sea of measurements. This challenge exemplifies a broader trend across sciences: as our ability to collect data grows exponentially, traditional statistical methods struggle to fully leverage this wealth of information. The core challenge lies in the inability of traditional methods to model complex probability distributions. These distributions are fundamental to describing natural phenomena, capturing the inherent uncertainties in measurements, the stochastic nature of system evolution, and the multiple solutions that may be consistent with observations. By learning these underlying distributions, we can solve a variety of scientifically relevant tasks, thereby accelerating scientific discovery.

Generative models are machine learning methods that learn probability distributions from samples. Once trained, they can generate synthetic data from the learned distribution. Often, the most interesting applications involve conditional generative models, which learn to generate outputs based on specific inputs. Familiar consumer examples include large language models like ChatGPT, which generate words conditional on preceding text, and text-to-image models such as Stable Diffusion or DALL-E, which generate images conditional on text prompts.

Scientific applications frequently involve input-output pairs from simulation data. These pairs might represent initial conditions and final outcomes, or simulation parameters and their resulting observations. Some applications, like climate forecasting, focus on generating outputs from inputs; here, a generative model can produce samples faster than running expensive simulations while preserving the natural variability in the system. In other cases, we aim to solve inverse problems: estimating initial conditions from final outcomes, or inferring system parameters from observations. By training on paired simulation data, we can build generative models that solve these inverse problems efficiently.

Consider cosmology: while we cannot experiment directly with universe evolution, we can simulate development under various cosmological models. A conditional generative model, trained on these simulations, can infer which combinations of parameters (like the Hubble constant, dark matter density, or universe curvature) most likely produced our observed universe. Similarly, in chemistry, while simulation programs can predict molecular properties from atomic structures, we often want the reverse: generating molecules with desired properties. A conditional generative model trained on simulation data can suggest

novel molecular structures optimized for specific characteristics like solubility or binding affinity.

The field has developed several approaches to meet these diverse needs. Current state-of-the-art methods fall into two main categories:

- Autoregressive models, which generate data one feature at a time (like ChatGPT generating text word-by-word)
- Transport models, which transform random noise into data through learned mappings (including diffusion models, variational autoencoders, and normalizing flows)

Among these, normalizing flows have found particular success in scientific applications, from astrophysics to particle physics to chemistry. Their appeal stems from three key properties: simple implementation, stable training, and exact likelihood computation (meaning they directly optimize the probability of generating the training data without approximations). This exactness, combined with fast training and generation, makes them especially valuable when computational resources are limited.

However, traditional normalizing flows face a significant constraint: they require carefully-designed invertible neural network architectures, which limits their expressiveness. This restriction makes it challenging to incorporate domain-specific knowledge or architectural innovations. For example, many molecular properties are invariant to rotation, but building rotation-equivariant normalizing flows is difficult under the invertibility constraint. Similar challenges arise when modeling data on non-Euclidean spaces, such as spherical data in Earth science or angular data in protein structure prediction.

This thesis advances generative modeling for scientific applications through two main contributions:

First, we introduce free-form flows, a novel family of generative models that preserve the benefits of normalizing flows while eliminating their architectural constraints. Through clever use of matrix calculus identities and automatic differentiation software, free-form flows enable exact maximum likelihood training with unrestricted neural networks, opening new possibilities for domain-specific architectures.

Second, we demonstrate practical applications of advanced machine learning models to particle physics, focusing on processing the massive data streams from the Large Hadron Collider. These applications showcase how models with rich representational spaces, including generative models, can accelerate the search for physics beyond the Standard Model.

Together, these contributions advance both the theoretical foundations and practical implementations of generative models, enabling their broader adoption across scientific disciplines. By removing key technical barriers and demonstrating concrete applications, this work aims to accelerate scientific discovery through more powerful and flexible modeling tools.

1.1 List of publications

In this section, I include the titles, author lists, and abstracts of the eight papers I published during my PhD.

The papers can be divided into two groups: (i) four machine learning papers in which I took a leading role (first authorship) in the project and (ii) four papers applying machine learning in particle physics in which my role was primarily supervision and project guidance.

1.1.1 Machine learning papers

As is the convention in machine learning, authors are ordered by the significance of their contributions. The first author typically leads the project, conducts the majority of the research, and writes much of the paper. The last author is typically the senior researcher or advisor who oversees the project or provides substantial guidance. Asterisks are used to indicate joint first authorship.

All four papers in this section are first-author or joint-first-author contributions. The first three concern the free-form flow, while the fourth is a topic in representation learning which does not appear in the thesis.

Lifting architectural constraints of injective flows

Peter Sorrenson*, Felix Draxler*, Armand Rousselot, Sander Hummerich, Lea Zimmermann, Ullrich Köthe

The Twelfth International Conference on Learning Representations (2024)

Abstract Normalizing Flows explicitly maximize a full-dimensional likelihood on the training data. However, real data is typically only supported on a lower-dimensional manifold leading the model to expend significant compute on modeling noise. Injective Flows fix this by jointly learning a manifold and the distribution on it. So far, they have been limited by restrictive architectures and/or high computational cost. We lift both constraints by a new efficient estimator for the maximum likelihood loss, compatible with free-form bottleneck architectures. We further show that naively learning both the data manifold and the distribution on it can lead to divergent solutions, and use this insight to motivate a stable maximum likelihood training objective. We perform extensive experiments on toy, tabular and image data, demonstrating the competitive performance of the resulting model.

Free-form flows: Make any architecture a normalizing flow

Felix Draxler*, Peter Sorrenson*, Lea Zimmermann, Armand Rousselot, Ullrich Köthe

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (2024)

Abstract Normalizing Flows are generative models that directly maximize the likelihood. Previously, the design of normalizing flows was largely constrained by the need for analytical invertibility. We overcome this constraint by a training procedure that

uses an efficient estimator for the gradient of the change of variables formula. This enables any dimension-preserving neural network to serve as a generative model through maximum likelihood training. Our approach allows placing the emphasis on tailoring inductive biases precisely to the task at hand. Specifically, we achieve excellent results in molecule generation benchmarks utilizing $E(n)$-equivariant networks at greatly improved sampling speed. Moreover, our method is competitive in an inverse problem benchmark, while employing off-the-shelf ResNet architectures. We publish our code at `https://github.com/vislearn/FFF`.

Learning distributions on manifolds with free-form flows

Peter Sorrenson*, Felix Draxler*, Armand Rousselot*, Sander Hummerich, Ullrich Köthe

Advances in Neural Information Processing Systems 37 (2024)

Abstract We propose Manifold Free-Form Flows (M-FFF), a simple new generative model for data on manifolds. The existing approaches to learning a distribution on arbitrary manifolds are expensive at inference time, since sampling requires solving a differential equation. Our method overcomes this limitation by sampling in a single function evaluation. The key innovation is to optimize a neural network via maximum likelihood on the manifold, possible by adapting the free-form flow framework to Riemannian manifolds. M-FFF is straightforwardly adapted to any manifold with a known projection. It consistently matches or outperforms previous single-step methods specialized to specific manifolds, and is competitive with multi-step methods with typically two orders of magnitude faster inference speed.

Learning distances from data with normalizing flows and score matching

Peter Sorrenson, Daniel Behrend-Uriarte, Christoph Schnörr, Ullrich Köthe

arXiv:2407.09297 (Under review)

Abstract Density-based distances (DBDs) offer an elegant solution to the problem of metric learning. By defining a Riemannian metric which increases with decreasing probability density, shortest paths naturally follow the data manifold and points are clustered according to the modes of the data. We show that existing methods to estimate Fermat distances, a particular choice of DBD, suffer from poor convergence in both low and high dimensions due to i) inaccurate density estimates and ii) reliance on graph-based paths which are increasingly rough in high dimensions. To address these issues, we propose learning the densities using a normalizing flow, a generative model with tractable density estimation, and employing a smooth relaxation method using a score model initialized from a graph-based proposal. Additionally, we introduce a dimension-adapted Fermat distance that exhibits more intuitive behavior when scaled to high dimensions and offers better numerical properties. Our work paves the way for practical use of density-based distances, especially in high-dimensional spaces.

1.1.2 Particle physics papers

As is the convention in particle physics, authors are listed alphabetically by last name. The ordering of authors has no other meaning.

In all four papers in this section, I did not lead the project or conduct the majority of the research. My role typically involved proposing promising project ideas and providing the machine learning know-how to see the project through to fruition.

Better latent spaces for better autoencoders

Barry M. Dillon, Tilman Plehn, Christof Sauer, Peter Sorrenson

SciPost Physics 11, 061 (2021)

Abstract Autoencoders as tools behind anomaly searches at the LHC have the structural problem that they only work in one direction, extracting jets with higher complexity but not the other way around. To address this, we derive classifiers from the latent space of (variational) autoencoders, specifically in Gaussian mixture and Dirichlet latent spaces. In particular, the Dirichlet setup solves the problem and improves both the performance and the interpretability of the networks.

Symmetries, safety, and self-supervision

Barry M. Dillon, Gregor Kasieczka, Hans Olischlager, Tilman Plehn, Peter Sorrenson, Lorenz Vogel

SciPost Physics 12, 188 (2022)

Abstract Collider searches face the challenge of defining a representation of high-dimensional data such that physical symmetries are manifest, the discriminating features are retained, and the choice of representation is new-physics agnostic. We introduce JetCLR to solve the mapping from low-level data to optimized observables though self-supervised contrastive learning. As an example, we construct a data representation for top and QCD jets using a permutation-invariant transformer-encoder network and visualize its symmetry properties. We compare the JetCLR representation with alternative representations using linear classifier tests and find it to work quite well.

A normalized autoencoder for LHC triggers

Barry M. Dillon, Luigi Favaro, Michael Krämer, Tilman Plehn, Peter Sorrenson

SciPost Physics Core 6, 074 (2023)

Abstract Autoencoders are an effective analysis tool for the LHC, as they represent one of its main goal of finding physics beyond the Standard Model. The key challenge is that out-of-distribution anomaly searches based on the compressibility of features do not apply to the LHC, while existing density-based searches lack performance. We present the first autoencoder which identifies anomalous jets symmetrically in the directions of higher and lower complexity. The normalized autoencoder combines a standard bottleneck architecture with a well-defined probabilistic description. It works better than all available autoencoders for top vs QCD jets and reliably identifies different dark-jet signals.

Jet diffusion versus JetGPT — modern networks for the LHC

Anja Butter, Nathan Huetsch, Sofia Palacios Schweitzer, Tilman Plehn, Peter Sorrenson, Jonas Spinner

arXiv:2305.10475 (Under review)

Abstract We introduce two diffusion models and an autoregressive transformer for LHC physics simulations. Bayesian versions allow us to control the networks and capture training uncertainties. After illustrating their different density estimation methods for simple toy models, we discuss their advantages for Z plus jets event generation. While diffusion networks excel through their precision, the transformer scales best with the phase space dimensionality. Given the different training and evaluation speed, we expect LHC physics to benefit from dedicated use cases for normalizing flows, diffusion models, and autoregressive transformers.

Chapter 2

Free-Form Flows

Normalizing flows have gained significant popularity in scientific applications due to their ability to provide exact likelihood training and their reliable performance on low-dimensional datasets. These characteristics make them particularly appealing to researchers and practitioners working with complex scientific problems. However, despite their advantages, normalizing flows suffer from a major limitation: their restrictive architectural design. This inflexibility poses a significant challenge when attempting to incorporate domain-specific knowledge or use novel architectures tailored to specific scientific problems. The constraints imposed by the normalizing flow framework often hinder the integration of cutting-edge techniques or problem-specific insights that could potentially enhance model performance and interpretability.

To address these limitations, this chapter introduces a new framework for building generative models for scientific applications. This approach retains the desirable properties of normalizing flows while removing the architectural constraints that have heretofore limited their adaptability. By doing so, it opens up new possibilities for researchers to design more flexible and powerful models that can better capture the intricacies of complex scientific phenomena, ultimately advancing the field of scientific machine learning.

Model development

This chapter covers work published in three papers (Draxler et al., 2024; Sorrenson et al., 2024b,a). The initial idea for the free-form flow emerged from my study of self-normalizing flows (Keller et al., 2021) and relative gradient optimization (Gresele et al., 2020). These works proposed networks structured such that the model weights directly contribute to the Jacobian calculation by using square weight matrices without skip connections. They then employ techniques to directly optimize the negative log-likelihood.

I realized that the main technique used, essentially Jacobi's formula from matrix calculus, could be applied to the Jacobian of the entire network. Furthermore, if an inverse function was known, one could use the Jacobian of the inverse function to make the whole process computationally tractable. I soon discovered that this framework generalized nicely to bottleneck models. At this point, I came across rectangular flows (Caterini et al., 2021), which had a similar concept, though they were less computationally efficient.

Improving upon rectangular flows formed the basis for the first paper, "Lifting architectural

constraints of injective flows" (Sorrenson et al., 2024b), with significant contributions from my co-author Felix Draxler and other co-authors, especially Armand Rousselot and Sander Hummerich. While working on that paper, we encountered substantial challenges in jointly training maximum likelihood and manifold learning.

As a next step, we shifted our focus to full-dimensional models, bringing the free-form flow back to its conceptual origins (Draxler et al., 2024). This significantly simplified both training and theory. My role in this paper was to expand the theoretical understanding of the model, while my colleagues conducted experiments.

Finally, discussions between myself, Felix Draxler, and Ullrich Köthe led to the idea of applying free-form flows to Riemannian manifolds such as spheres and tori (Sorrenson et al., 2024a). I suggested that simply wrapping the encoder and decoder in a projection function would probably work. Felix developed a plausible estimator, implemented it, and found it to be effective. I then provided a more solid theoretical foundation for the idea, based on Riemannian geometry, while the other co-authors conducted extensive experiments.

Although the development of free-form flows did not follow a linear path, I will present them in an order that facilitates understanding:

1. **Full-dimensional version:** This provides the strongest theoretical foundation and is conceptually simpler.
2. **Bottleneck version:** Our initial breakthrough, which generalizes the full-dimensional case.
3. **Application to Riemannian manifolds:** A subsequent extension that broadens the applicability to complex geometries.

2.1 Full-dimensional free-form flows

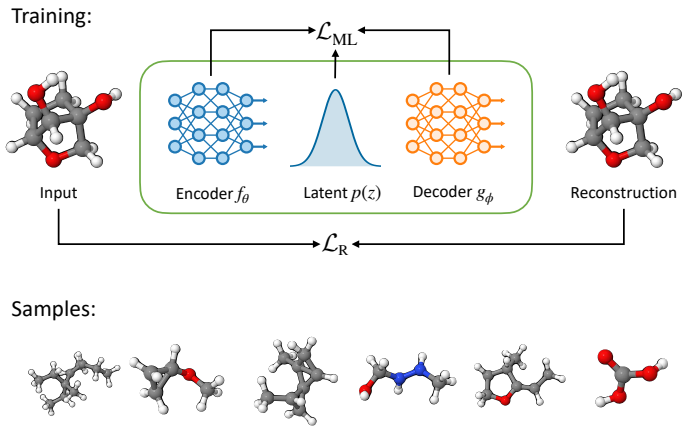


Figure 2.1: *Free-form flows (FFF) train a pair of encoder and decoder neural networks with a fast maximum likelihood estimator $\mathcal{L}_{\mathrm{ML}}$ and reconstruction loss $\mathcal{L}_{\mathrm{R}}$. This enables the training of any dimension-preserving architecture as a one-step generative model. For example, an equivariant graph neural network can be trained on the QM9 dataset to generate molecules by predicting atom positions and properties in a single decoder evaluation.* (Bottom) *Stable molecules sampled from our $E(3)$-FFF trained on the QM9 dataset for several molecule sizes.*

This section is based on work previously published as "Free-form flows: Make any architecture a normalizing flow" at AISTATS 2024 (Draxler et al., 2024). All figures are reproduced from that publication unless stated otherwise.

2.1.1 Normalizing flows

To provide the necessary background for free-form flows, we first need to discuss normalizing flows.

Normalizing flows (Rezende and Mohamed, 2015; Kobyzev et al., 2021; Papamakarios et al., 2021) are generative models that learn a probability density via a parameterized diffeomorphism f. The probability density is obtained through the change of variables formula:

$$p(x) = p_Z(f(x))\,|f'(x)| \tag{2.1}$$

where p_Z is a simple latent distribution and where $|f'(x)|$ is the absolute value of the determinant of the Jacobian matrix of the transformation. $|f'(x)|$ is also known as the Jacobian of f.

Sampling is achieved by first drawing a sample from the latent distribution p_Z and then applying the inverse transformation f^{-1} to map it back to the data space.

A common strategy to design normalizing flows is through coupling blocks (Dinh et al., 2015, 2017; Durkan et al., 2019), where f is the composition of a series of blocks, each with a triangular Jacobian. This structure makes the determinant easy to calculate, making the above formula tractable.

Normalizing flows are trained by minimizing the negative log-likelihood on a training set:

$$\mathcal{L} = \mathbb{E}_{p_{\mathrm{data}}(x)}[-\log p(x)] \tag{2.2}$$

2.1.2 Free-form flows compared to normalizing flows

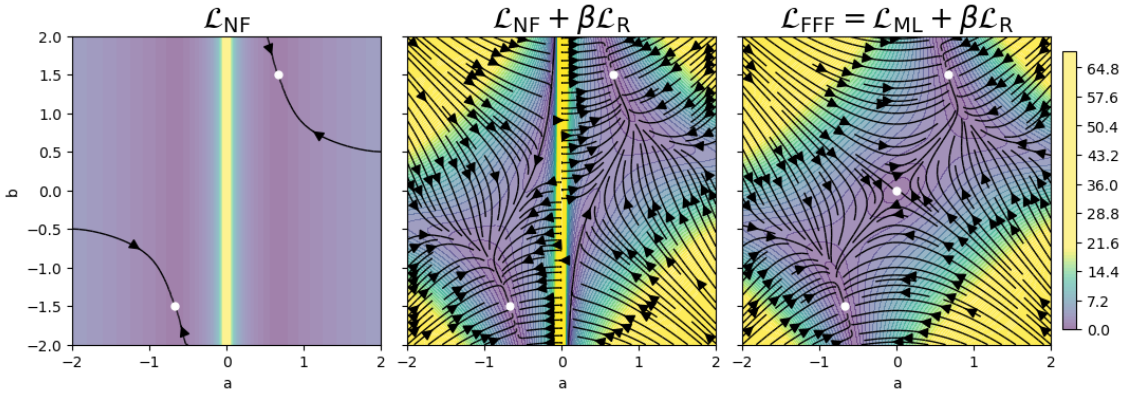


Figure 2.2: *Gradient landscapes for a normalizing flow* (left)*, a normalizing flow with reconstruction loss* (center)*, and a free-form flow* (right) *in a linear 1D model with* $f(x) = ax$*,* $g(z) = bz$*,* $\beta = 1$*, and a zero-mean Gaussian data distribution with standard deviation* 1.5*. Flow lines indicate gradient direction, and contours represent gradient magnitude. White dots mark critical points. We plot gradient magnitudes instead of loss contours, as only gradients are meaningful for the FFF loss.*
The transformation from normalizing flows to free-form flows is shown from left to right. Initially, only diffeomorphisms are considered (constrained to curves $b = a^{-1}$*). The addition of a reconstruction loss extends the optimization to the full* a*-*b *space. Finally, the normalizing flow loss* $\mathcal{L}_{\text{NF}}$ *is replaced by the FFF maximum likelihood estimator* $\mathcal{L}_{\text{ML}}$*, avoiding the costly computation of the log-Jacobian and its gradient.*
The minima of $\mathcal{L}_{\text{FFF}}$ *are the same as those of* $\mathcal{L}_{\text{NF}}$ *at* $(\pm 2/3, \pm 1.5)$*, with an additional saddle point at* $a = b = 0$*, which is not a convergence point in practice. Thus, optimizing the free-form flow yields the same solutions as the equivalent normalizing flow.*
Adapted from Figure 2 in Draxler et al. (2024).

Normalizing flows are designed to satisfy two constraints in order to make them suitable for maximum likelihood training:

1. **Exact invertibility.** The change of variables formula relies on the exact invertibility of f for its validity.
2. **Tractable Jacobian.** The Jacobian $|f'(x)|$ is used in the change of variables formula, so this quantity should not be too computationally expensive.

These requirements typically lead to the design of relatively constrained architectures, such as coupling blocks (Dinh et al., 2015, 2017; Durkan et al., 2019), or to the use of continuous-time models (Neural ODEs (Chen et al., 2018; Grathwohl et al., 2019)).

The free-form flow is designed to address both constraints without resorting to particular architectural requirements – the models used are "free-form," meaning they can be any neural network (as long as the input and output dimensions are equal). The solutions are as follows:

1. **Approximate invertibility.** Instead of using a single, specially parameterized model for both forward and backward transformations, we employ two separate models with independent weights, coupled by a reconstruction loss. We demonstrate that optimizing over a broader class of functions (Lipschitz continuous) using the normalizing flow loss combined with a sufficiently weighted reconstruction term

yields solutions equivalent to those obtained by optimizing the normalizing flow loss alone over diffeomorphisms (Theorem 2.1.8).

2. **Tractable Jacobian approximation.** The log-Jacobian is expensive to evaluate in general, but only its gradient with respect to model parameters is needed for optimization via gradient descent. This gradient can be tractably approximated by combining derivatives from the forward and backward models (Theorem 2.1.4), and the approximation results in the same optima as using the exact Jacobian (Theorem 2.1.10). For this reason, we can substitute our maximum likelihood surrogate loss $\mathcal{L}_{\mathrm{ML}}$ (detailed below) in place of the normalizing flow loss $\mathcal{L}_{\mathrm{NF}}$.

Figure 2.2 demonstrates these modifications: here the gradient landscape of a linear model is visualized, showing how optimization changes under (i) the addition of a reconstruction loss, and (ii) the use of $\mathcal{L}_{\mathrm{ML}}$ in place of $\mathcal{L}_{\mathrm{NF}}$.

In this way, the free-form flow can be viewed as a relaxation of normalizing flow training, with the advantage that optimization is performed over a larger function class (dimension-preserving, Lipschitz continuous neural networks rather than some parameterization of diffeomorphisms). Importantly, this alleviates design constraints, making it easier for practitioners to adapt new methods to their problems.

In the rest of the section, I will state and prove the theorems that justify the validity of the free-form flow method, followed by additional theoretical results that deepen the understanding of the method, including an interpretation of the model as a form of variational autoencoder (VAE).

2.1.3 Preliminary definitions

We need to discuss automatic differentiation and the stop-gradient operation, which are essential tools in the implementation of free-form flows.

Vector-Jacobian and Jacobian-vector products Automatic differentiation (AD) libraries (such as PyTorch, TensorFlow, or JAX) implement vector-Jacobian products as low-level primitives. This means that products of the form $v^T f'(x)$ can be calculated from a backward pass through the computational graph of the function f. For this reason, vector-Jacobian products (VJPs) are known as "backward-mode AD." More recently, these libraries have also offered Jacobian-vector products (JVPs) of the form $f'(x)v$, which are computed in parallel with the output of the function (i.e., during the forward pass). This type of computation is therefore known as "forward-mode AD." In PyTorch, VJPs and JVPs are conveniently implemented in the `torch.func` library.

Definition 2.1.1 (Stop-gradient)**.** *The stop-gradient operation* `SG` *is a computational operation that sets the gradients of its input to zero, effectively detaching the input from the computation graph:*

$$\mathtt{SG}[x] = x, \qquad \frac{\mathrm{d}}{\mathrm{d}x}\mathtt{SG}[x] = 0 \tag{2.3}$$

This operation deviates from the standard rules of calculus and should be viewed as a computational flag rather than a proper function. It is particularly useful for computing

gradients of the form $\nabla(\texttt{SG}[a(x)]b(x)) = a(x)\nabla b(x)$, which cannot be directly calculated using standard calculus. The stop-gradient operation is implemented in all deep learning libraries, such as `torch.Tensor.detach()` in PyTorch.

Now we are ready to define some loss functions.

Definition 2.1.2 (Free-form flow loss functions)**.**

$$\mathcal{L}_{\text{NF}}[f](x) = -\log p_Z(f(x)) - \log|f'(x)| \tag{2.4}$$
$$\mathcal{L}_{\text{ML}}[f,g](x,v) = -\log p_Z(f(x)) - \texttt{SG}\left[v^T g'(f(x))\right] f'(x)v \tag{2.5}$$
$$\mathcal{L}_{\text{R}}[f,g](x) = \|g(f(x)) - x\|^2 \tag{2.6}$$
$$\mathcal{L}_{\text{FFF}}[f,g](x,v) = \mathcal{L}_{\text{ML}}[f,g](x,v) + \beta\mathcal{L}_{\text{R}}[f,g](x) \tag{2.7}$$

The loss functions are defined per sample x and per random vector v in the case of the FFF loss. We can also define overall losses such as $\mathcal{L}_{\text{FFF}}[f,g] = \mathbb{E}_{x,v}[\mathcal{L}_{\text{FFF}}[f,g](x,v)]$, where x is always sampled from the training distribution p_{data}.

Definition 2.1.3 (Well-behaved probability distribution)**.** *In the context of this thesis, we call a probability distribution defined on* $\mathbb{R}^n$ well-behaved *if it satisfies the following properties:*

- *Absolutely continuous: This means that the distribution assigns zero probability to any set with zero Lebesgue measure. In practical terms, it ensures that the distribution has a well-defined density function.*
- *Full support: The distribution assigns positive probability to every open subset of* $\mathbb{R}^n$*.*
- *Continuous density: The probability density function of the distribution is continuous on* $\mathbb{R}^n$*.*

2.1.4 Free-form flow estimator

To prove the theorems, we first need to establish some intermediate results, which are well-known.

Lemma 2.1.1 (Jacobi's formula)**.** *Let* $A : \mathbb{R} \to \mathbb{R}^{n\times n}$ *be a matrix-valued function depending on the scalar variable* t*. Then*

$$\frac{\mathrm{d}}{\mathrm{d}t}|A(t)| = |A(t)|\,\mathrm{tr}\left(A(t)^{-1}\frac{\mathrm{d}A(t)}{\mathrm{d}t}\right) \tag{2.8}$$

Equivalently:

$$\frac{\mathrm{d}}{\mathrm{d}t}\log|A(t)| = \mathrm{tr}\left(A(t)^{-1}\frac{\mathrm{d}A(t)}{\mathrm{d}t}\right) \tag{2.9}$$

The proof can be found in standard textbooks on matrix calculus (Magnus and Neudecker, 2019, Theorem 8.1) and on Wikipedia.[1] The second expression follows directly from the definition of the derivative of the logarithm: $\frac{\mathrm{d}}{\mathrm{d}t}\log f(t) = \frac{1}{f(t)}\frac{\mathrm{d}}{\mathrm{d}t}f(t)$.

[1] `https://en.wikipedia.org/wiki/Jacobi%27s_formula`

Lemma 2.1.2 (Hutchinson trace estimator (Hutchinson, 1989)). *Let $A \in \mathbb{R}^{n \times n}$ be a matrix and v a random variable such that $\mathbb{E}_v[vv^T] = \mathbb{I}$. Then*

$$\mathbb{E}_v[v^T A v] = \operatorname{tr}(A) \tag{2.10}$$

Proof. Use the cyclic property and linearity of the trace:

$$\mathbb{E}_v[v^T A v] = \mathbb{E}_v[\operatorname{tr}(v^T A v)] = \mathbb{E}_v[\operatorname{tr}(A v v^T)] = \operatorname{tr}(A \mathbb{E}_v[vv^T]) = \operatorname{tr}(A) \tag{2.11}$$

□

Lemma 2.1.3 (Inverse Jacobian is Jacobian of inverse). *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be a diffeomorphism. Then*

$$f'(x)^{-1} = f^{-1\prime}(f(x)) \tag{2.12}$$

Proof. Consider differentiating $f^{-1} \circ f$, which is just the identity function:

$$\begin{aligned} \frac{\partial}{\partial x} f^{-1}(f(x)) &= \frac{\partial f^{-1}(f(x))}{\partial f(x)} \cdot \frac{\partial f(x)}{\partial x} && (2.13) \\ &= f^{-1\prime}(f(x)) \cdot f'(x) && (2.14) \\ &= \mathbb{I} && (2.15) \end{aligned}$$

Multiplying by $f'(x)^{-1}$ from the right leads to the result. □

We now present the main result of this section: the gradient of the negative log-likelihood objective can be efficiently estimated using $\mathcal{L}_{\mathrm{ML}}$, provided we have access to the inverse of the forward transformation f.

Theorem 2.1.4 (Free-form flow estimator). *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be a diffeomorphism. Let p_Z and p_{data} be well-behaved probability distributions on $\mathbb{R}^n$, and let p be the pushforward of p_Z by f^{-1}. Let $v \in \mathbb{R}^n$ be a random variable such that $\mathbb{E}_v[vv^T] = \mathbb{I}$. Then for all $x \in \mathbb{R}^n$, $\mathcal{L}_{\mathrm{ML}}[f, f^{-1}]$ has the same gradient as the negative log-likelihood:*

$$\frac{\mathrm{d}}{\mathrm{d}t} \mathcal{L}_{\mathrm{ML}}[f, f^{-1}] = \frac{\mathrm{d}}{\mathrm{d}t} \mathbb{E}_x \left[-\log p(x) \right] \tag{2.16}$$

where f depends on t and x is sampled from p_{data}. In particular, if f is parameterized by θ, the gradient with respect to θ is the same:

$$\nabla_\theta \mathcal{L}_{\mathrm{ML}}[f_\theta, f_\theta^{-1}] = \nabla_\theta \mathbb{E}_x \left[-\log p_\theta(x) \right] \tag{2.17}$$

Proof. We know from Section 2.1.1 that

$$\log p(x) = \log p_Z(f(x)) + \log |f'(x)| \tag{2.18}$$

Suppose that f depends implicitly on a parameter t. By applying Lemma 2.1.1 to

the log-determinant, we find

$$\frac{\mathrm{d}}{\mathrm{d}t}\log|f'(x)| = \mathrm{tr}\left(f'(x)^{-1}\frac{\mathrm{d}}{\mathrm{d}t}f'(x)\right) \tag{2.19}$$

to which we can apply Lemma 2.1.3:

$$\frac{\mathrm{d}}{\mathrm{d}t}\log|f'(x)| = \mathrm{tr}\left(f^{-1\prime}(z)\frac{\mathrm{d}}{\mathrm{d}t}f'(x)\right) \tag{2.20}$$

where $z = f(x)$. Substituting this result into Equation (2.18) gives

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\log p(x) &= \frac{\mathrm{d}}{\mathrm{d}t}\log p_Z(f(x)) + \mathrm{tr}\left(f^{-1\prime}(z)\frac{\mathrm{d}}{\mathrm{d}t}f'(x)\right) && (2.21)\\
&= \frac{\mathrm{d}}{\mathrm{d}t}\log p_Z(f(x)) + \mathbb{E}_v\left[v^T f^{-1\prime}(z)\frac{\mathrm{d}}{\mathrm{d}t}f'(x)v\right] && (2.22)\\
&= \frac{\mathrm{d}}{\mathrm{d}t}\log p_Z(f(x)) + \frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}_v\left[\mathtt{SG}[v^T f^{-1\prime}(z)]f'(x)v\right] && (2.23)\\
&= -\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}_v\left[\mathcal{L}_{\mathrm{ML}}[f, f^{-1}](x, v)\right] && (2.24)
\end{aligned}$$

where we used Hutchinson's trace estimator (Lemma 2.1.2) and the stop-gradient operation (Definition 2.1.1). By taking the expectation over x, we obtain:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{L}_{\mathrm{ML}}[f, f^{-1}] = \frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}_x\left[-\log p(x)\right] \tag{2.25}$$

This proves the first part of the theorem.
To extend this result to the gradient with respect to θ, we note that the parameter t can be thought of as any component of θ. Since the equality holds for the derivative with respect to any such component, it also holds for the gradient with respect to the entire parameter vector θ:

$$\nabla_\theta\mathcal{L}_{\mathrm{ML}}[f_\theta, f_\theta^{-1}] = \nabla_\theta\mathbb{E}_x\left[-\log p_\theta(x)\right] \tag{2.26}$$

This completes the proof of the theorem. □

2.1.5 Relaxing the invertibility requirement

In this section, we show that jointly optimizing a normalizing flow loss and a reconstruction loss yields equivalent results to optimizing the normalizing flow loss alone. Furthermore, we demonstrate that by incorporating a reconstruction loss with sufficient weight, we can extend the optimization domain beyond diffeomorphisms to include any Lipschitz continuous function. This key insight is illustrated visually in Figure 2.2.

We first prove several lemmas, which lead to the main result in Theorem 2.1.8.

Lemma 2.1.5 (Normalizing flow loss learns the training density)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be a diffeomorphism with inverse f^{-1}. Let p_Z and p_{data} be well-behaved probability*

distributions on $\mathbb{R}^n$ and let p be the pushforward of p_Z by f^{-1}. Suppose that f minimizes $\mathcal{L}_{\text{NF}}[f]$. Then p is well-behaved, and its density function is equal to p_{data} almost everywhere.

Proof. First, let's express the loss function $\mathcal{L}_{\text{NF}}[f]$ in terms of the Kullback-Leibler (KL) divergence between p_{data} and p:

$$\begin{aligned}
\mathcal{L}_{\text{NF}}[f] &= \mathbb{E}_x[-\log p_Z(f(x)) - \log|f'(x)|] && (2.27)\\
&= \mathbb{E}_x[-\log p(x)] \quad \text{(by change of variables formula)} && (2.28)\\
&= \mathbb{E}_x[-\log p(x) + \log p_{\text{data}}(x) - \log p_{\text{data}}(x)] && (2.29)\\
&= \mathcal{D}_{\text{KL}}(p_{\text{data}}\|p) + \mathbb{E}_x[-\log p_{\text{data}}(x)] && (2.30)
\end{aligned}$$

Note that $\mathbb{E}_x[-\log p_{\text{data}}(x)]$ is the entropy of p_{data}, which is constant with respect to f. Therefore, minimizing $\mathcal{L}_{\text{NF}}[f]$ is equivalent to minimizing $\mathcal{D}_{\text{KL}}(p_{\text{data}}\|p)$.
The KL divergence has two important properties:

1. $\mathcal{D}_{\text{KL}}(p_{\text{data}}\|p) \geq 0$ for all distributions p_{data} and p.
2. $\mathcal{D}_{\text{KL}}(p_{\text{data}}\|p) = 0$ if and only if $p_{\text{data}} = p$ almost everywhere.

When f minimizes $\mathcal{L}_{\text{NF}}[f]$, it also minimizes $\mathcal{D}_{\text{KL}}(p_{\text{data}}\|p)$. Given that $\mathcal{D}_{\text{KL}}(p_{\text{data}}\|p) \geq 0$, the minimum value it can achieve is 0. This minimum is achieved when $p_{\text{data}} = p$ almost everywhere.
Now, we need to show that p is well-behaved:

1. Absolutely continuous: Since p_Z is absolutely continuous and f is a diffeomorphism, p is also absolutely continuous.
2. Full support: p_Z has full support and f is a diffeomorphism, so p also has full support on $\mathbb{R}^n$.
3. Continuous density: The density of p is given by $p(x) = p_Z(f(x))|f'(x)|$. Since p_Z is continuous, f is smooth (as a diffeomorphism), and the determinant is a continuous function of matrix entries, p is continuous.

Therefore, p is well-behaved and its density function is equal to p_{data} almost everywhere. □

Lemma 2.1.6. *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be a Lipschitz continuous function, and let p_Z be a well-behaved probability density function. Then the normalizing flow loss $\mathcal{L}_{\text{NF}}$ is bounded from below.*

Proof. Recall that the normalizing flow loss is given by:

$$\mathcal{L}_{\text{NF}} = \mathbb{E}_x\left[-\log p_Z(f(x)) - \log|f'(x)|\right] \tag{2.31}$$

First, consider the term $-\log p_Z(f(x))$. Since p_Z is well-behaved, it is bounded above by some constant M. Therefore, $-\log p_Z(f(x)) \geq -\log M$ for all x.
Next, consider the term $-\log|f'(x)|$. Since f is Lipschitz continuous, there exists a constant $L > 0$ such that $\|f'(x)\|_2 \leq L$ for all x, where $\|\cdot\|_2$ denotes the spectral norm. As $|f'(x)|$ is the product of the singular values of $f'(x)$, and the spectral norm is the maximum singular value, we have $|f'(x)| \leq L^n$. Therefore, $-\log|f'(x)| \geq -n\log L$ for all x.

Combining these bounds, we have:

$$\mathcal{L}_{\text{NF}} \geq \mathbb{E}_x \left[-\log M - n \log L \right] = -\log M - n \log L \tag{2.32}$$

Thus, $\mathcal{L}_{\text{NF}}$ is bounded from below by the constant $-\log M - n \log L$. □

Lemma 2.1.7. *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^n$ be continuous functions that minimize the reconstruction loss $\mathcal{L}_R[f, g] = \mathbb{E}_x[\|x - g(f(x))\|^2]$, where x is drawn from a distribution with density p_{data}. Then f is injective on the support of p_{data}, g is surjective onto the support of p_{data}, and g is the inverse of f on the image of f restricted to the support of p_{data}.*

Proof. Suppose f and g minimize $\mathcal{L}_R[f, g]$. This means $\mathcal{L}_R[f, g] = 0$, as the reconstruction loss is non-negative and achieves its minimum at zero.
For $\mathcal{L}_R[f, g] = 0$, we must have:

$$x = g(f(x)) \quad \forall x \in \text{support}(p_{\text{data}}) \tag{2.33}$$

From this equation, we can conclude:

- f is injective (one-to-one) on the support of p_{data}: If $f(x_1) = f(x_2)$ for some $x_1, x_2 \in \text{support}(p_{\text{data}})$, then $g(f(x_1)) = g(f(x_2))$, which implies $x_1 = x_2$.
- g is surjective (onto) onto the support of p_{data}: Since $x = g(f(x))$ for all $x \in \text{support}(p_{\text{data}})$, the range of g includes the entire support of p_{data}.
- g is the inverse of f on $f(\text{support}(p_{\text{data}}))$: For any $y \in f(\text{support}(p_{\text{data}}))$, there exists an $x \in \text{support}(p_{\text{data}})$ such that $f(x) = y$. Then $g(y) = g(f(x)) = x$, showing that g inverts f on $f(\text{support}(p_{\text{data}}))$.

Therefore, f is injective on the support of p_{data}, g is surjective onto the support of p_{data}, and g is the inverse of f on the image of f restricted to the support of p_{data}. □

Theorem 2.1.8 (Reconstruction loss induces invertibility)**.** *Consider the optimization of $\mathcal{L}_{\text{NF}}[f] + \beta \mathcal{L}_R[f, g]$ over Lipschitz continuous functions f (where f is not necessarily a diffeomorphism) and C^1 functions g. Suppose that the training distribution p_{data} and latent prior p_Z are well-behaved (and hence have full support). Then, there exists a critical value $\beta_{\text{crit}} \geq 0$ such that for any $\beta > \beta_{\text{crit}}$, all global minimizers (f^*, g^*) of this combined loss function satisfy the following conditions:*

1. *f^* is a diffeomorphism*
2. *$g^* = (f^*)^{-1}$*
3. *f^* is a global minimizer of $\mathcal{L}_{\text{NF}}[f]$ restricted to diffeomorphisms*

Proof. We will prove this theorem in two steps: first, we'll show that for sufficiently large β, any global minimizer (f^*, g^*) must satisfy $\mathcal{L}_R[f^*, g^*] = 0$. Then, we'll show that this implies the three conditions stated in the theorem.
Step 1: Existence of β_{crit}.
Let $L_0^* = \inf_{f,g:\mathcal{L}_R[f,g]=0} \mathcal{L}_{\text{NF}}[f]$ be the infimum of $\mathcal{L}_{\text{NF}}$ over all pairs (f, g) that

achieve perfect reconstruction. L_0^* is finite by Lemma 2.1.6.
For any pair (f, g), the combined loss $\mathcal{L}_{\text{NF}}[f] + \beta\mathcal{L}_R[f, g]$ is linear in β with a non-negative slope $\mathcal{L}_R[f, g]$.
For pairs (f, g) where $\mathcal{L}_R[f, g] \neq 0$, there exists a β large enough such that their combined loss exceeds L_0^*. Specifically, for:

$$\beta > \max\left(0, \frac{L_0^* - \mathcal{L}_{\text{NF}}[f]}{\mathcal{L}_R[f, g]}\right) \tag{2.34}$$

the combined loss for (f, g) will be greater than L_0^*.
Define β_{crit} as:

$$\beta_{\text{crit}} = \sup_{f,g:\mathcal{L}_R[f,g]\neq 0} \max\left(0, \frac{L_0^* - \mathcal{L}_{\text{NF}}[f]}{\mathcal{L}_R[f, g]}\right) \tag{2.35}$$

For any $\beta > \beta_{\text{crit}}$, all global minimizers (f^*, g^*) of the combined loss must satisfy $\mathcal{L}_R[f^*, g^*] = 0$. If not, there would exist a pair (f, g) with $\mathcal{L}_R[f, g] = 0$ that achieves a lower combined loss, contradicting the assumption of global minimization.
Step 2: Implications of $\mathcal{L}_R[f^*, g^*] = 0$.

1. By Lemma 2.1.7, since p_{data} has full support (as it is well-behaved), f^* is injective on $\mathbb{R}^n$, g^* is surjective onto $\mathbb{R}^n$, and g^* is the inverse of f^* on the image of f^*.
   Since p_Z is also well-behaved and thus has full support, f^* must be surjective onto $\mathbb{R}^n$ to minimize $\mathcal{L}_{\text{NF}}[f^*]$. If it weren't, there would be regions of the latent space with zero probability under $f^*_{\#}p_{\text{data}}$, leading to infinite $\mathcal{L}_{\text{NF}}[f^*]$. Combining the injectivity and surjectivity of f^*, we conclude that f^* is bijective. Since f^* is also Lipschitz continuous, it is a diffeomorphism.
2. From point 1, we already have that $g^* = (f^*)^{-1}$.
3. Finally, since f^* is a diffeomorphism, minimizing $\mathcal{L}_{\text{NF}}[f^*] + \beta\mathcal{L}_R[f^*, g^*]$ is equivalent to minimizing $\mathcal{L}_{\text{NF}}[f^*]$ over diffeomorphisms. Therefore, f^* is a global minimizer of $\mathcal{L}_{\text{NF}}[f]$ restricted to diffeomorphisms.

Thus, we have shown that for any $\beta > \beta_{\text{crit}}$, all global minimizers (f^*, g^*) satisfy the three conditions stated in the theorem. □

The above theorem and proof can be generalized to the case where p_{data} does not have full support. In this scenario, we need to modify our requirements:

1. f should be Lipschitz continuous only over $\text{support}(p_{\text{data}})$, rather than the entire $\mathbb{R}^n$.
2. f need not be Lipschitz continuous or even continuous outside $\text{support}(p_{\text{data}})$.
3. g should be defined as the inverse of f on $f(\text{support}(p_{\text{data}}))$.

With these modifications, the theorem still holds, ensuring that for sufficiently large β:

1. f is bijective from $\text{support}(p_{\text{data}})$ to $f(\text{support}(p_{\text{data}}))$.
2. $g = f^{-1}$ on $f(\text{support}(p_{\text{data}}))$.
3. f minimizes $\mathcal{L}_{\text{NF}}[f]$ among all such functions.

2.1.6 Error bound

The following theorem bounds the error introduced when approximating the inverse of the encoder Jacobian matrix with the decoder Jacobian matrix. The bounds demonstrate that this approximation is accurate when the encoder and decoder are well-matched, with the error controlled by their degree of mismatch as measured by $\|f'g' - \mathbb{I}\|_F$.

Theorem 2.1.9. *Let f and g be C^1 functions, and let f' and g' denote the Jacobians of f at x and g at $f(x)$, respectively (we use this shorthand throughout the theorem). Suppose that f is locally invertible at x, meaning f' is an invertible matrix. Let $\|\cdot\|_F$ denote the Frobenius norm of a matrix and let ∇ denote the derivative with respect to a parameter of f. Then:*

1. *The absolute difference between $\nabla \log |f'|$ and its trace-based approximation is bounded:*

$$|\mathrm{tr}((\nabla f')g') - \nabla \log |f'|| \leq \|(\nabla f')(f')^{-1}\|_F \|f'g' - \mathbb{I}\|_F \tag{2.36}$$

2. *If we extend the local invertibility of f to invertibility wherever $p_{\text{data}}(x)$ has support, then the difference in gradients between $\mathcal{L}_{\text{ML}}$ and $\mathcal{L}_{\text{NF}}$ is bounded:*

$$|\nabla \mathcal{L}_{\text{ML}} - \nabla \mathcal{L}_{\text{NF}}| \leq \mathbb{E}_x \left[\|(\nabla f')(f')^{-1}\|_F^2\right]^{\frac{1}{2}} \mathbb{E}_x \left[\|f'g' - \mathbb{I}\|_F^2\right]^{\frac{1}{2}} \tag{2.37}$$

Proof. We begin by recalling two key inequalities:

1. The Cauchy-Schwarz inequality for an inner product $\langle \cdot, \cdot \rangle$:

$$|\langle u, v \rangle|^2 \leq \langle u, u \rangle \langle v, v \rangle \tag{2.38}$$

2. The trace forms the Frobenius inner product over matrices: $\langle A, B \rangle_F = \mathrm{tr}(A^T B)$. Applying Cauchy-Schwarz to this inner product yields:

$$\begin{aligned} |\,\mathrm{tr}(A^T B)|^2 &\leq \mathrm{tr}(A^T A)\,\mathrm{tr}(B^T B) && (2.39)\\ &= \|A\|_F^2 \|B\|_F^2 && (2.40) \end{aligned}$$

   where $\|A\|_F = \sqrt{\mathrm{tr}(A^T A)}$ is the Frobenius norm of A.

Now, let's prove the two parts of the theorem:

Part 1. Recall Jacobi's formula (Lemma 2.1.1):

$$\nabla \log |f'| = \mathrm{tr}((\nabla f')(f')^{-1}) \tag{2.41}$$

We can now derive the bound:

$$\begin{aligned} |\mathrm{tr}((\nabla f')g') - \nabla \log |f'|| &= \left|\mathrm{tr}((\nabla f')g') - \mathrm{tr}((\nabla f')(f')^{-1})\right| && (2.42)\\ &= \left|\mathrm{tr}((\nabla f')(g' - (f')^{-1}))\right| && (2.43)\\ &= \left|\mathrm{tr}((\nabla f')(f')^{-1}(f'g' - \mathbb{I}))\right| && (2.44)\\ &\leq \|(\nabla f')(f')^{-1}\|_F \|f'g' - \mathbb{I}\|_F && (2.45) \end{aligned}$$

where the last line applies the Cauchy-Schwarz inequality.

Part 2. For this part, we need two additional inequalities:

3. Jensen's inequality for a convex function $\alpha : \mathbb{R} \to \mathbb{R}$:

$$\alpha(\mathbb{E}[x]) \leq \mathbb{E}[\alpha(x)] \tag{2.46}$$

4. Hölder's inequality (with $p = q = 2$) for random variables X and Y:

$$\mathbb{E}[|XY|] \leq \mathbb{E}[|X|^2]^{\frac{1}{2}} \mathbb{E}[|Y|^2]^{\frac{1}{2}} \tag{2.47}$$

Now, we can derive the bound:

$$\begin{aligned}
|\nabla \mathcal{L}_{\text{ML}} - \nabla \mathcal{L}_{\text{NF}}| &= |\mathbb{E}_x \left[\text{tr}((\nabla f')g')\right] - \mathbb{E}_x \left[\nabla \log |f'|\right]| && (2.48)\\
&= \left|\mathbb{E}_x \left[\text{tr}((\nabla f')(f')^{-1}(f'g' - I))\right]\right| && (2.49)\\
&\leq \mathbb{E}_x \left[\left|\text{tr}((\nabla f')(f')^{-1}(f'g' - I))\right|\right] && (2.50)\\
&\leq \mathbb{E}_x \left[\|(\nabla f')(f')^{-1}\|_F \|f'g' - I\|_F\right] && (2.51)\\
&\leq \mathbb{E}_x \left[\|(\nabla f')(f')^{-1}\|_F^2\right]^{\frac{1}{2}} \mathbb{E}_x \left[\|f'g' - I\|_F^2\right]^{\frac{1}{2}} && (2.52)
\end{aligned}$$

where we apply Jensen's inequality, then Cauchy-Schwarz, and finally Hölder's inequality. □

2.1.7 Equivalence of free-form flow to normalizing flow

This key theorem establishes that any critical point of the normalizing flow objective is also a critical point of the free-form flow loss. While additional critical points may exist, as illustrated in Figure 2.2, we hypothesize that these points can be easily identified by their nonzero reconstruction error. Note that we discuss critical points rather than minima because the FFF loss, due to its use of stop-gradient, is not a proper loss function and only its gradients are meaningful.

Theorem 2.1.10 (Equivalence of critical points)**.** *Let f^* be a minimizer of $\mathcal{L}_{\text{NF}}[f]$ over diffeomorphisms and let $g^* = (f^*)^{-1}$. Suppose that f^* is parameterized by θ and that $\nabla_{\theta_i} f^{*\prime}(x)$ is bounded for all x and all i. Then (f^*, g^*) is a critical point of $\mathcal{L}_{\text{FFF}}$.*

Proof. By part 2 of Theorem 2.1.9, at (f^*, g^*) we have:

$$|\nabla_{\theta_i} \mathcal{L}_{\text{ML}} - \nabla_{\theta_i} \mathcal{L}_{\text{NF}}| \leq \mathbb{E}_x \left[\|(\nabla_{\theta_i} f^{*\prime})(f^{*\prime})^{-1}\|_F^2\right]^{\frac{1}{2}} \mathbb{E}_x \left[\|f^{*\prime} g^{*\prime} - \mathbb{I}\|_F^2\right]^{\frac{1}{2}} \tag{2.53}$$

where $f^{*\prime} = f^{*\prime}(x)$, $g^{*\prime} = g^{*\prime}(f^*(x))$.
Since $\nabla_{\theta_i} f^{*\prime}$ is bounded by assumption and $(f^{*\prime})^{-1}$ is bounded because f^* is a diffeomorphism, the first term $\mathbb{E}_x \left[\|(\nabla_{\theta_i} f^{*\prime})(f^{*\prime})^{-1}\|_F^2\right]^{\frac{1}{2}}$ is finite. The second term $\mathbb{E}_x \left[\|f^{*\prime} g^{*\prime} - \mathbb{I}\|_F^2\right]^{\frac{1}{2}}$ is zero since $f^{*\prime} g^{*\prime} = \mathbb{I}$. Therefore, the right-hand side of the equation is zero, implying that the gradients of $\mathcal{L}_{\text{ML}}$ and $\mathcal{L}_{\text{NF}}$ are equal and thus zero at (f^*, g^*).
Given that the reconstruction loss is also at a minimum, (f^*, g^*) is a critical point of $\mathcal{L}_{\text{FFF}}$. □

2.1.8 Links between FFF and VAE

Here we establish a connection between free-form flows (FFFs) and variational autoencoders (VAEs). Specifically, we demonstrate that the loss function $\mathcal{L}_{\mathrm{NF}} + \beta\mathcal{L}_{\mathrm{R}}$ can be interpreted as a VAE loss function. This interpretation retains the familiar VAE decoder while allowing for a more flexible class of VAE encoders.

To begin, let us introduce the necessary notation and provide some context for our analysis.

Our generative model is as follows:

$$p(z) = \mathcal{N}(z; 0, I) \tag{2.54}$$
$$p_\phi(x|z) = \delta(x - g_\phi(z)) \tag{2.55}$$

meaning that to generate data we sample from a standard normal latent distribution and pass the sample through the generator network g_ϕ. The corresponding inference model is:

$$q(x) = \text{data distribution} \tag{2.56}$$
$$q_\theta(z|x) = \delta(z - f_\theta(x)) \tag{2.57}$$

Our goal is to minimize the KL divergence

$$\mathcal{D}_{\mathrm{KL}}(q(x)\|p_\phi(x)) = \mathbb{E}_{q(x)}\left[\log \frac{q(x)}{p_\phi(x)}\right] \tag{2.58}$$
$$= \mathbb{E}_{q(x)}\left[-\log \int p_\phi(x, z)\,\mathrm{d}z\right] - h(q(x)) \tag{2.59}$$

where h denotes the differential entropy. Unfortunately, this divergence is intractable due to the integral over z (though it would be tractable if g_ϕ^{-1} and $\log|J_{g_\phi}(z)|$ are tractable due to the change of variables formula – in this case, the model would be a typical normalizing flow). The variational autoencoder (VAE) is a latent variable model that solves this problem by minimizing the divergence over the joint x and z space. This is an upper bound to the divergence over just x:

$$\mathcal{D}_{\mathrm{KL}}(q_\theta(x, z)\|p_\phi(x, z)) = E_{q_\theta(x,z)}\left[\log \frac{q_\theta(x, z)}{p_\phi(x, z)}\right] \tag{2.60}$$
$$= E_{q_\theta(x,z)}\left[\log \frac{q(x)}{p_\phi(x)} + \log \frac{q_\theta(z|x)}{p_\phi(z|x)}\right] \tag{2.61}$$
$$= \mathcal{D}_{\mathrm{KL}}(q(x)\|p_\phi(x)) + E_{q(x)}\left[\mathcal{D}_{\mathrm{KL}}(q_\theta(z|x)\|p_\phi(z|x))\right] \tag{2.62}$$
$$\geq \mathcal{D}_{\mathrm{KL}}(q(x)\|p_\phi(x)) \tag{2.63}$$

The inequality comes from the fact that KL divergences are always non-negative. Unfortunately, this KL divergence is not well-defined due to the delta distributions, which make the joint distributions over x and z degenerate. Unless the support of $q_\theta(x, z)$ and $p_\phi(x, z)$ exactly overlap, which is very unlikely for arbitrary f_θ and g_ϕ, the divergence will be infinite.

The solution is to introduce an auxiliary variable $\tilde{x}$ which is the data with some added Gaussian noise:

$$p(\tilde{x}|x) = q(\tilde{x}|x) = \mathcal{N}(\tilde{x}; x, \sigma^2 I) \tag{2.64}$$

The generative model over z and $\tilde{x}$ is therefore

$$p(z) = \mathcal{N}(z; 0, I) \tag{2.65}$$
$$p_\phi(\tilde{x}|z) = \mathcal{N}(\tilde{x}; g_\phi(z), \sigma^2 I) \tag{2.66}$$

and the inference model is

$$q(\tilde{x}) = \int q(x)q(\tilde{x}|x)\,\mathrm{d}x \tag{2.67}$$
$$q(\tilde{x}|x) = \mathcal{N}(\tilde{x}; x, \sigma^2 I) \tag{2.68}$$
$$q_\theta(z|\tilde{x}) = \frac{\int q(x)q(\tilde{x}|x)q_\theta(z|x)\,\mathrm{d}x}{\int q(x)q(\tilde{x}|x)\,\mathrm{d}x} \tag{2.69}$$

Note that $q_\theta(z|\tilde{x})$ is extremely flexible in comparison to typical VAE variational posteriors where often a Gaussian with diagonal covariance is used.

Now the relationship between $\tilde{x}$ and z has become stochastic, and we can safely minimize the KL divergence, which will always take on finite values:

$$\mathcal{D}_{\mathrm{KL}}(q_\theta(\tilde{x}, z)\|p_\phi(\tilde{x}, z)) \geq \mathcal{D}_{\mathrm{KL}}(q(\tilde{x})\|p_\phi(\tilde{x})) \tag{2.70}$$

The KL divergence between the noised variables is known as a spread KL divergence $\tilde{\mathcal{D}}_{\mathrm{KL}}$ (Zhang et al., 2020):

$$\tilde{\mathcal{D}}_{\mathrm{KL}}(q(x)\|p_\phi(x)) = \mathcal{D}_{\mathrm{KL}}(q(\tilde{x})\|p_\phi(\tilde{x})) \tag{2.71}$$

Theorem 2.1.11 (Link between FFF and VAE)**.** *Let f_θ be a diffeomorphism and g_ϕ be C^1. Define the spread KL divergence $\tilde{\mathcal{D}}_{\mathrm{KL}}$ as the KL divergence between distributions convolved with isotropic Gaussian noise of variance σ^2. Let $\beta = 1/(2\sigma^2)$ and*

$$D = \mathcal{D}_{\mathrm{KL}}(q(\tilde{x}, z)\|p(\tilde{x}, z)) \tag{2.72}$$

Then the following equalities hold:

$$\nabla_\theta(\mathcal{L}_{\mathrm{NF}} + \beta\mathcal{L}_{\mathrm{R}}) = \nabla_\theta D \quad \text{and} \quad \nabla_\phi(\mathcal{L}_{\mathrm{NF}} + \beta\mathcal{L}_{\mathrm{R}}) = \nabla_\phi D \tag{2.73}$$

Moreover, we have the inequality:

$$D \geq \tilde{\mathcal{D}}_{\mathrm{KL}}(q(x)\|p_\phi(x)) \tag{2.74}$$

Consequently, minimizing $\mathcal{L}_{\mathrm{NF}} + \beta\mathcal{L}_{\mathrm{R}}$ is equivalent to minimizing an upper bound on $\tilde{\mathcal{D}}_{\mathrm{KL}}(q(x)\|p_\phi(x))$.

Proof. We begin with the identity for differential entropy (Papoulis and Pillai, 2002):

$$h(Z) = h(X) + \mathbb{E}[\log|f'(X)|] \tag{2.75}$$

where h is the differential entropy, and $Z = f(X)$ with f being invertible. Applying

this to our context, we have:

$$h(q(z|\tilde{x})) = h(q(x|\tilde{x})) + \mathbb{E}_{q(x|\tilde{x})}[\log|f'(x)|] = \mathbb{E}_{q(x)}[\log|f'(x)|] + \text{const.} \quad (2.76)$$

For clarity, we'll omit θ and ϕ subscripts. Let ϵ be a standard normal variable. We can now derive D step by step.
First, we start with the KL divergence definition:

$$\begin{aligned} D &= \mathbb{E}_{q(\tilde{x},z)}\left[\log q(\tilde{x}) + \log q(z|\tilde{x}) - \log p(z) - \log p(\tilde{x}|z)\right] && (2.77)\\ &= \mathbb{E}_{q(\tilde{x})}\left[-h(q(z|\tilde{x}))\right] + \mathbb{E}_{q(\tilde{x},z)}\left[-\log p(z) - \log p(\tilde{x}|z)\right] + \text{const.} && (2.78)\end{aligned}$$

Using Equation (2.76) and treating $h(q(x|\tilde{x}))$ as constant:

$$D = \mathbb{E}_{q(x)}\left[-\log|f'(x)|\right] + \mathbb{E}_{q(\tilde{x},z)}\left[-\log p(z) - \log p(\tilde{x}|z)\right] + \text{const.} \quad (2.79)$$

Next, we change variables from $\tilde{x}, z$ to x, ϵ with $\tilde{x} = x + \sigma\epsilon$ and $z = f(x)$:

$$\begin{aligned} D =& \mathbb{E}_{q(x)q(\epsilon)}\left[-\log|f'(x)| - \log p_Z(f(x)) - \log p(\tilde{X} = x + \sigma\epsilon|Z = f(x))\right] \\ &+ \text{const.} \end{aligned} \quad (2.80)$$

Substituting the Gaussian log-likelihood for $p(\tilde{x}|z)$ and discarding constants:

$$D = \mathbb{E}_{q(x)q(\epsilon)}\left[-\log|f'(x)| - \log p_Z(f(x)) + \frac{1}{2\sigma^2}\|x + \sigma\epsilon - g(f(x))\|^2\right] + \text{const.} \quad (2.81)$$

Expanding the quadratic term and separating the ϵ-dependent part:

$$\begin{aligned} D = \mathbb{E}_{q(x)q(\epsilon)}\Big[&-\log|f'(x)| - \log p_Z(f(x)) + \frac{1}{2\sigma^2}\|x - g(f(x))\|^2 \\ &+ \frac{1}{\sigma}\epsilon^T(x - g(f(x)))\Big] + \text{const.} \end{aligned} \quad (2.82)$$

Finally, evaluating the ϵ expectation (noting $\mathbb{E}[\epsilon] = 0$) and setting $\beta = 1/(2\sigma^2)$:

$$D = \mathbb{E}_{q(x)}\left[-\log|f'(x)| - \log p_Z(f(x)) + \beta\|x - g(f(x))\|^2\right] + \text{const.} \quad (2.83)$$

Note that this final expression equals $\mathcal{L}_{\text{NF}} + \beta\mathcal{L}_{\text{R}}$ up to the constant. Since the constant doesn't depend on θ or ϕ, we have:

$$\nabla_\theta(\mathcal{L}_{\text{NF}} + \beta\mathcal{L}_{\text{R}}) = \nabla_\theta D \quad \text{and} \quad \nabla_\phi(\mathcal{L}_{\text{NF}} + \beta\mathcal{L}_{\text{R}}) = \nabla_\phi D \quad (2.84)$$

We have already established that:

$$D \geq \mathcal{D}_{\text{KL}}(q(\tilde{x})\|p_\phi(\tilde{x})) = \tilde{\mathcal{D}}_{\text{KL}}(q(x)\|p_\phi(x)) \quad (2.85)$$

In conclusion, the gradients of $\mathcal{L}_{\text{NF}} + \beta\mathcal{L}_{\text{R}}$ provide an unbiased estimate of D's gradients. Thus, minimizing $\mathcal{L}_{\text{NF}} + \beta\mathcal{L}_{\text{R}}$ under stochastic gradient descent converges to the same solutions as minimizing D, which upper bounds the spread KL divergence between $q(x)$ and $p_\phi(x)$. □

2.2 Free-form injective flows

This section is based on work previously published as "Lifting architectural constraints of injective flows" at ICLR 2024 (Sorrenson et al., 2024b). All figures are reproduced from that publication.

This section generalizes the free-form flow to bottleneck models, meaning models with a latent space of lower dimensionality than the data space.

2.2.1 Background

This section provides essential mathematical background for working with autoencoding models. While most results presented here are well-established in the literature, we introduce several key definitions: consistent autoencoders, unique projection domains, and functional pseudoinverses.

The key result of this section is Theorem 2.2.3, a result commonly used in the literature that generalizes the change of variables formula to the bottleneck setting.

Definition 2.2.1 (Consistent autoencoder)**.** *Let $f : \mathbb{R}^D \to \mathbb{R}^d$ be an encoder that compresses data to a latent space and let $g : \mathbb{R}^d \to \mathbb{R}^D$ be a decoder that decompresses the latent representation. A full-dimensional model has $d = D$ while a bottleneck model has $d < D$. If $f \circ g : \mathbb{R}^d \to \mathbb{R}^d$ is the identity, then we call f and g* consistent*. For example, the forward and inverse function of a normalizing flow are consistent as $f^{-1} = g$.*

Definition 2.2.2 (Matrix pseudoinverse)**.** *The* pseudoinverse *(also known as Moore-Penrose inverse) of a matrix $A \in \mathbb{R}^{m \times n}$, denoted A^+, is the unique matrix that satisfies the following four conditions:*

1. $AA^+A = A$
2. $A^+AA^+ = A^+$
3. $(AA^+)^T = AA^+$
4. $(A^+A)^T = A^+A$

If A has full rank, then $A^+ = (A^TA)^{-1}A^T$ if $m > n$ or $A^+ = A^T(AA^T)^{-1}$ if $m < n$. If A is square ($m = n$), then $A^+ = A^{-1}$.

Definition 2.2.3 (Unique projection domain)**.** *Let $g : \mathbb{R}^d \to \mathbb{R}^D$ with $D > d$ be C^1 and injective. The* unique projection domain *of g, denoted $\mathcal{U}_g$, is the set of points in $\mathbb{R}^D$ for which the projection onto the image of g is uniquely defined. Formally,*

$$\mathcal{U}_g = \{x \in \mathbb{R}^D : \exists! y \in \mathbb{R}^d \text{ such that } \|g(y) - x\| = \inf_{z \in \mathbb{R}^d} \|g(z) - x\|\} \tag{2.86}$$

where $\exists!$ means "there exists a unique."

Lemma 2.2.1. *The complement of $\mathcal{U}_g$ has measure zero.*

Proof. Let $d(x) = \inf_{z \in \mathbb{R}^d} \|g(z) - x\|$ be the distance function from a point $x \in \mathbb{R}^D$ to the image of g.

1. **Lipschitz continuity of d:** Let $x_1, x_2 \in \mathbb{R}^D$. Let $z_2 \in \mathbb{R}^d$ be such that $\|x_2 - g(z_2)\| = d(x_2)$. Then by the triangle inequality:

$$d(x_1) \leq \|x_1 - g(z_2)\| \leq \|x_1 - x_2\| + \|x_2 - g(z_2)\| = \|x_1 - x_2\| + d(x_2) \quad (2.87)$$

   Since we could swap the roles of x_1 and x_2, we must have:

$$|d(x_1) - d(x_2)| \leq \|x_1 - x_2\| \quad (2.88)$$

   which shows that d is Lipschitz continuous.
2. **Differentiability of d almost everywhere:** By Rademacher's theorem, any Lipschitz continuous function is differentiable almost everywhere. Therefore, d is differentiable almost everywhere in $\mathbb{R}^D$.
3. **$\mathcal{U}_g$ is the set where d is differentiable:** By definition, $\mathcal{U}_g$ is the set of points in $\mathbb{R}^D$ for which the projection onto the image of g is uniquely defined. This is precisely the set where the distance function d is differentiable. Therefore, $\mathcal{U}_g$ is the set where d is differentiable.

Since d is differentiable almost everywhere, the set where d is not differentiable (i.e., the complement of $\mathcal{U}_g$) has measure zero. □

Definition 2.2.4 (Functional pseudoinverse)**.** *Let $g : \mathbb{R}^d \to \mathbb{R}^D$ with $D > d$ be C^1 and injective. The* functional pseudoinverse *$f : \mathcal{U}_g \to \mathbb{R}^d$ of g is the unique function that minimizes the reconstruction loss $\mathcal{L}_{\mathrm{R}} = \mathbb{E}_x \left[\|g(f(x)) - x\|^2\right]$, where x is sampled from a well-behaved probability distribution p_{data}.*

Theorem 2.2.2 (Properties of the functional pseudoinverse)**.** *Let $g : \mathbb{R}^d \to \mathbb{R}^D$ with $D > d$ be C^1 and injective, and let $f : \mathcal{U}_g \to \mathbb{R}^d$ be its functional pseudoinverse. Then:*

1. *f is unique.*
2. *f and g are consistent, i.e., $f \circ g$ is the identity on $\mathbb{R}^d$.*
3. *For all x in the image of g, $f'(x) = g'(f(x))^+$, where $g'(f(x))^+$ is the matrix pseudoinverse of $g'(f(x))$.*

Proof.

1. For any $x \in \mathcal{U}_g$, there exists a unique $y \in \mathbb{R}^D$ such that $\|y - x\| = \inf_{y' \in \mathbb{R}^D} \|y' - x\|$. Since g is injective, there is a unique $z \in \mathbb{R}^d$ such that $g(z) = y$. Therefore, $f(x) = z = \arg\min_{z' \in \mathbb{R}^d} \|g(z') - x\|$ is uniquely defined for all $x \in \mathcal{U}_g$, proving the uniqueness of f.
2. Let $z \in \mathbb{R}^d$. By definition of f, we have:

$$f(g(z)) = \arg\min_{z' \in \mathbb{R}^d} \|g(z') - g(z)\| \quad (2.89)$$

   Since g is injective, the unique minimizer of this expression is $z' = z$. Therefore, $f(g(z)) = z$ for all $z \in \mathbb{R}^d$, which means $f \circ g$ is the identity on

$\mathbb{R}^d$.

3. We use the calculus of variations. The reconstruction loss $\mathcal{L}_{\text{R}}$ has the form:

$$\mathcal{L}_{\text{R}} = \int p_{\text{data}}(x)\|g(f(x)) - x\|^2 \, dx \tag{2.90}$$

By the Euler-Lagrange equations, we have:

$$p_{\text{data}}(x)(g(f(x)) - x)_k g'_{ki}(f(x)) = 0 \tag{2.91}$$

where we use Einstein notation to imply a sum over k. Dividing by p_{data} and differentiating with respect to x_j:

$$g'_{kl}(f(x))f'_{lj}(x)g'_{ki}(f(x)) - \delta_{kj}g'_{ki}(f(x)) + (g(f(x)) - x)_k g''_{kij}(f(x)) = 0 \tag{2.92}$$

Evaluating on the image of g, using $g(f(x)) = x$:

$$g'(f(x))^T g'(f(x))f'(x) = g'(f(x))^T \tag{2.93}$$

Therefore, $f'(x) = [g'(f(x))^T g'(f(x))]^{-1} g'(f(x))^T = g'(f(x))^+$, which is the matrix pseudoinverse of $g'(f(x))$.

□

Definition 2.2.5 (Matrix volume)**.** *The* volume *of a matrix $A \in \mathbb{R}^{m\times n}$, denoted* $\text{vol}(A)$*, is defined as the product of its singular values. If σ_i are the singular values of A, then*

$$\text{vol}(A) = \prod_{i=1}^{\min(m,n)} \sigma_i \tag{2.94}$$

The volume can also be defined in terms of the determinant. Specifically, for a "fat" matrix $A \in \mathbb{R}^{m\times n}$ with $m < n$, we use

$$\text{vol}(A) = \sqrt{|AA^T|} \tag{2.95}$$

and for a "skinny" matrix $A \in \mathbb{R}^{m\times n}$ with $m > n$, we use

$$\text{vol}(A) = \sqrt{|A^T A|} \tag{2.96}$$

If $m = n$, then the volume of the matrix A is simply the absolute value of its determinant:

$$\text{vol}(A) = |A| \tag{2.97}$$

Theorem 2.2.3 (Change of variables across dimensions)**.** *Let $g : \mathbb{R}^d \to \mathbb{R}^D$ be an injective C^1 function and let $f : \mathbb{R}^D \to \mathbb{R}^d$ be its functional pseudoinverse. Then, the change of variable formula generalizes to:*

$$p_X(x) = p_Z(f(x)) \, \text{vol}(g'(f(x)))^{-1} \tag{2.98}$$

$$= p_Z(f(x)) \, \text{vol}(f'(\hat{x})) \tag{2.99}$$

where $\hat{x} = g(f(x))$. Note that this expression only integrates to 1 if we restrict integration to the image of g. Therefore, it should be regarded as defining a probability distribution on this manifold, not in the ambient space $\mathbb{R}^D$.

Proof. The change of variables formula describes how probability densities transform under an injective "pushforward" function g. We start by deriving this formula for the case when $d = D$, meaning g is an invertible function. Let p_Z be a base density and p_X the pushforward density obtained by mapping samples from p_Z through g. We can express this as:

$$p_X(x) = \int p(x|z)p_Z(z)\,\mathrm{d}z \tag{2.100}$$

$$= \int \delta(x - g(z))p_Z(z)\,\mathrm{d}z \tag{2.101}$$

$$= \int \delta(x - \hat{x})p_Z(f(\hat{x}))|g'(f(\hat{x}))|^{-1}\,\mathrm{d}\hat{x} \tag{2.102}$$

$$= p_Z(f(x))|g'(f(x))|^{-1} \tag{2.103}$$

Here, we used the change of variables $\hat{x} = g(z)$, implying $z = f(\hat{x})$ and $|g'(z)|\,\mathrm{d}z = \mathrm{d}\hat{x}$, with f being the inverse of g.
Now, consider the case where g maps from $\mathbb{R}^d$ to $\mathbb{R}^D$ with $d < D$. We generalize the change of variables $\hat{x} = g(z)$ using $z = f(\hat{x})$ and $\mathrm{vol}(g'(z))\,\mathrm{d}z = \mathrm{d}\hat{x}$, where f and g are consistent (see Chapter 5 of Krantz and Parks (2008)). This gives us:

$$p_X(x) = \int \delta(x - g(z))p_Z(z)\,\mathrm{d}z \tag{2.104}$$

$$= \int \delta(x - \hat{x})p_Z(f(\hat{x}))\,\mathrm{vol}(g'(f(\hat{x})))^{-1}\,\mathrm{d}\hat{x} \tag{2.105}$$

This expression defines a probability density in the full ambient space $\mathbb{R}^D$ (albeit a degenerate distribution), but we cannot easily remove the integral. However, we can convert it into an expression resembling the full-dimensional case, but defined only on the image of g, by restricting integration to the image of g:

$$p_X(x) = p_Z(f(x))\,\mathrm{vol}(g'(f(x)))^{-1} \tag{2.106}$$

Finally, we note that, due to Theorem 2.2.2, $f'(\hat{x}) = g'(f(x))^+$ and since $\mathrm{vol}(A^+) = \mathrm{vol}(A)^{-1}$ for a full rank matrix, we can also write

$$p_X(x) = p_Z(f(x))\,\mathrm{vol}(f'(\hat{x})) \tag{2.107}$$

□

2.2.2 Joint maximum likelihood and manifold learning

In order to train a model that simultaneously learns a manifold and the density on it, we define a training objective in two parts:

1. Reconstruction loss to bring data close to a low-dimensional manifold.
2. Maximum likelihood on the manifold.

This results in the loss:

$$\mathcal{L}_{\text{NF}}^{\text{on}} + \beta \mathcal{L}_{\text{R}} \tag{2.108}$$

where we use the NF (normalizing flow) subscript to specify an exact maximum likelihood loss and indicate that it is evaluated on the manifold, i.e., the image of g.

Using Theorem 2.2.3, the maximum likelihood part of the loss has the form:

$$\mathcal{L}_{\text{NF}}^{\text{on}} = \mathbb{E}_x \left[-\log p(\hat{x}) \right] = \mathbb{E}_x \left[-\log p_Z(f(x)) + \log \operatorname{vol}(g'(f(x))) \right] \tag{2.109}$$

where f is the functional pseudoinverse of g and $\hat{x} = g(f(x))$ is the projection onto the manifold.

Again by Theorem 2.2.3, this loss can alternatively be defined in terms of the encoder Jacobian $f'(\hat{x})$:

$$\mathcal{L}_{\text{NF}}^{\text{on}} = \mathbb{E}_x \left[-\log p_Z(f(x)) - \log \operatorname{vol}(f'(\hat{x})) \right] \tag{2.110}$$

This form makes the loss more similar to the full-dimensional normalizing flow loss and to the free-form flow loss used in the previous section. As in free-form flows, we can approximate this by an unbiased estimator (see Theorem 2.2.5, proven after the following lemma):

$$\mathcal{L}_{\text{ML}}^{\text{on}} = \mathbb{E}_{x,v} \left[-\log p_Z(f(x)) - v^T f'(\hat{x}) \texttt{SG}[g'(f(x))v] \right] \tag{2.111}$$

Lemma 2.2.4 (Generalization of Jacobi's formula to matrix volume)**.** *Let $A \in \mathbb{R}^{D \times d}$ be a full-rank matrix with $d \leq D$. Then*

$$\frac{\mathrm{d}}{\mathrm{d}t} \log \operatorname{vol}(A) = \operatorname{tr}\left(A^{+} \frac{\mathrm{d}A}{\mathrm{d}t} \right) \tag{2.112}$$

Proof. Suppose without loss of generality that A is $m \times n$ with $m \geq n$. By definition, $\operatorname{vol}(A) = \sqrt{|A^T A|}$. Taking the logarithm, we have

$$\log \operatorname{vol}(A) = \frac{1}{2} \log |A^T A|. \tag{2.113}$$

Differentiating with respect to t, we get

$$\frac{\mathrm{d}}{\mathrm{d}t} \log \operatorname{vol}(A) = \frac{1}{2} \frac{\mathrm{d}}{\mathrm{d}t} \log |A^T A|. \tag{2.114}$$

Using the matrix logarithm derivative formula, we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t} \log |A^T A| = \operatorname{tr}\left((A^T A)^{-1} \frac{\mathrm{d}(A^T A)}{\mathrm{d}t} \right). \tag{2.115}$$

Since $\frac{\mathrm{d}(A^T A)}{\mathrm{d}t} = A^T \frac{\mathrm{d}A}{\mathrm{d}t} + \left(\frac{\mathrm{d}A}{\mathrm{d}t}\right)^T A$, we have

$$\frac{\mathrm{d}}{\mathrm{d}t} \log |A^T A| = \operatorname{tr}\left((A^T A)^{-1} \left(A^T \frac{\mathrm{d}A}{\mathrm{d}t} + \left(\frac{\mathrm{d}A}{\mathrm{d}t}\right)^T A \right) \right). \tag{2.116}$$

Using that $\operatorname{tr}(B + B^T) = 2\operatorname{tr}(B)$ for any square matrix B, we get

$$\frac{\mathrm{d}}{\mathrm{d}t} \log |A^T A| = 2 \operatorname{tr}\left((A^T A)^{-1} A^T \frac{\mathrm{d}A}{\mathrm{d}t} \right). \tag{2.117}$$

Noting that $(A^T A)^{-1} A^T = A^+$, we finally obtain

$$\frac{\mathrm{d}}{\mathrm{d}t} \log \operatorname{vol}(A) = \operatorname{tr}\left(A^+ \frac{\mathrm{d}A}{\mathrm{d}t} \right). \tag{2.118}$$

□

Theorem 2.2.5 (Free-form injective flow estimator (on-manifold))**.** *Let $g : \mathbb{R}^d \to \mathbb{R}^D$ be a C^1 injective function and let $f : \mathbb{R}^D \to \mathbb{R}^d$ be its functional pseudoinverse. Let p_Z be a well-behaved probability distribution on $\mathbb{R}^d$ and let p be the pushforward of p_Z by g. Let $v \in \mathbb{R}^d$ be a random variable such that $\mathbb{E}_v[vv^T] = \mathbb{I}$. Denote the image of g as $g(\mathbb{R}^d)$. Then for all $x \in g(\mathbb{R}^d)$, $\mathcal{L}^{\text{on}}_{\text{ML}}$ has the same gradient as the negative log-likelihood:*

$$\frac{\mathrm{d}}{\mathrm{d}t} \mathcal{L}^{\text{on}}_{\text{ML}}[f, g] = \frac{\mathrm{d}}{\mathrm{d}t} \mathbb{E}_x \left[-\log p(\hat{x}) \right] \tag{2.119}$$

In particular, if g (and by extension f) are parameterized by θ, the gradient with respect to θ is the same:

$$\nabla_\theta \mathcal{L}^{\text{on}}_{\text{ML}}[f_\theta, g_\theta] = \nabla_\theta \mathbb{E}_x \left[-\log p_\theta(\hat{x}) \right]. \tag{2.120}$$

Proof. The logic is the same as in the proof of Theorem 2.1.4, but we use the pseudoinverse instead of the inverse, utilizing Lemma 2.2.4.

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t} \log p(\hat{x}) &= \frac{\mathrm{d}}{\mathrm{d}t} \log p_Z(f(x)) + \operatorname{tr}\left(f'(\hat{x})^+ \frac{\mathrm{d}}{\mathrm{d}t} f'(\hat{x}) \right) && (2.121)\\
&= \frac{\mathrm{d}}{\mathrm{d}t} \log p_Z(f(x)) + \operatorname{tr}\left(\left(\frac{\mathrm{d}}{\mathrm{d}t} f'(\hat{x}) \right) g'(f(x)) \right) && (2.122)\\
&= \frac{\mathrm{d}}{\mathrm{d}t} \log p_Z(f(x)) + \mathbb{E}_v \left[v^T \left(\frac{\mathrm{d}}{\mathrm{d}t} f'(\hat{x}) \right) g'(f(x)) v \right] && (2.123)\\
&= \frac{\mathrm{d}}{\mathrm{d}t} \log p_Z(f(x)) + \frac{\mathrm{d}}{\mathrm{d}t} \mathbb{E}_v \left[v^T f'(\hat{x}) \texttt{SG}[g'(f(x)) v] \right] && (2.124)
\end{aligned}$$

Therefore,

$$\frac{\mathrm{d}}{\mathrm{d}t} \mathbb{E}_x \left[-\log p(\hat{x}) \right] = \frac{\mathrm{d}}{\mathrm{d}t} \mathcal{L}^{\text{on}}_{\text{ML}}[f, g] \tag{2.125}$$

This naturally extends to the gradient with respect to θ:

$$\nabla_\theta \mathcal{L}^{\text{on}}_{\text{ML}}[f_\theta, g_\theta] = \nabla_\theta \mathbb{E}_x \left[-\log p_\theta(\hat{x}) \right] \tag{2.126}$$

□

2.2.3 Maximum likelihood in bottleneck models

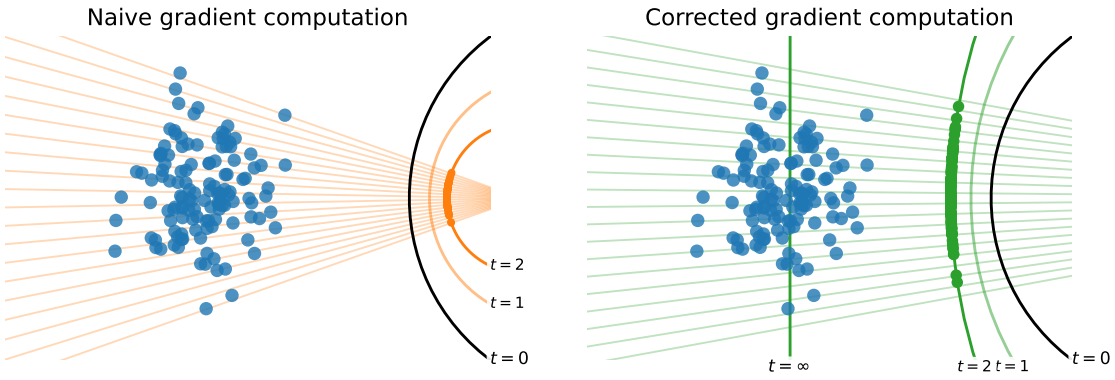


Figure 2.3: *Naive training of autoencoders with negative log-likelihood (NLL) can lead to pathological solutions* (left). *Starting with the initialization* ($t = 0$, black)*, gradient steps can increase the curvature of the learned manifold* ($t = 1, 2$, orange)*. This reduces NLL because the entropy of the projected data decreases as the points move closer to one another. This effect is stronger than that of the reconstruction loss. We address this problem by evaluating the volume change off the manifold* (right)*. This adjustment moves the manifold closer to the data and reduces its curvature* ($t = 1, 2$, green)*, until it eventually centers the manifold on the data with zero curvature* ($t = \infty$, green)*. Light lines indicate the set of points that map to the same latent point. Data is projected onto the manifold at* $t = 2$.

In the previous section, we discussed training models with a combination of a reconstruction and a likelihood term. We might ask what happens if we only train with the likelihood term, yielding a loss function reminiscent of normalizing flows. Using the exact maximum likelihood variant, our loss is:

$$\mathcal{L}^{\text{on}}_{\text{NF}}(x) = -\log p_Z(f(x)) - \log \text{vol}(f'(\hat{x})). \tag{2.127}$$

Unfortunately, optimizing this loss can lead us to learn a degenerate decoder manifold, an issue raised in Brehmer and Cranmer (2020). Here we expand on their argument.

First, consider that if f and g are consistent, then $f(\hat{x}) = f(g(f(x))) = f(x)$ and the per-sample loss is invariant to projections: $\mathcal{L}^{\text{on}}_{\text{NF}}(\hat{x}) = \mathcal{L}^{\text{on}}_{\text{NF}}(x)$. This means that we can write our loss as:

$$\mathcal{L}^{\text{on}}_{\text{NF}} = \mathbb{E}_{p_{\text{data}}(x)}[\mathcal{L}^{\text{on}}_{\text{NF}}(x)] = \mathbb{E}_{\hat{p}_{\text{data}}(\hat{x})}[\mathcal{L}^{\text{on}}_{\text{NF}}(\hat{x})], \tag{2.128}$$

where $\hat{p}_{\text{data}}(\hat{x})$ is the probability density of the projection of the training data onto the decoder manifold. Now consider that the negative log-likelihood loss is one part of a KL divergence, and KL divergences are always non-negative:

$$\mathcal{D}_{\text{KL}}(\hat{p}_{\text{data}}(\hat{x}) \| p_\theta(\hat{x})) = -H(\hat{p}_{\text{data}}(\hat{x})) - \mathbb{E}_{\hat{p}_{\text{data}}(\hat{x})}[\log p_\theta(\hat{x})] \geq 0. \tag{2.129}$$

As a result, the loss is lower bounded by the entropy of the data projected onto the manifold:

$$\mathcal{L}^{\text{on}}_{\text{NF}} = -\mathbb{E}_{\hat{p}_{\text{data}}(\hat{x})}[\log p_\theta(\hat{x})] \geq H(\hat{p}_{\text{data}}(\hat{x})). \tag{2.130}$$

Unlike in standard normalizing flow optimization, where the right-hand side would be fixed, here the entropy depends on the projection learned by the model. Thus, the model could modify the projection such that entropy is as low as possible. We break this pathology down into two cases:

1. **A model manifold that does not align with the data manifold but instead intersects it.** For example, Brehmer and Cranmer (2020) discuss a case where a linear model learns to project a data distribution to a single point on the manifold, thus reducing its entropy to $-\infty$, the lowest possible value. To the best of our knowledge, this can be fixed by adding noise and a reconstruction loss with sufficiently high weight. Appendix C in Sorrenson et al. (2024b) contains a proof that characterizes the solutions for linear models, which are the same as principal component analysis (PCA) if $\beta \geq 1/2\sigma^2$, where σ^2 is the smallest eigenvalue of the data covariance matrix.

2. **A model manifold that concentrates the data by using high curvature.** See Figure 2.3 *(left)*. This newly identified pathological case only occurs in nonlinear models; hence Brehmer and Cranmer (2020) did not notice this effect in their linear example. Importantly, this is **not** fixed by adding a reconstruction loss.

Most existing injective flows avoid this by using a two-stage training process, which first learns a projection and then the distribution of the projected data in the latent space. To enable jointly learning a manifold and a maximum-likelihood density on it, we need to find a fix for the pathology.

2.2.4 Towards a well-behaved loss

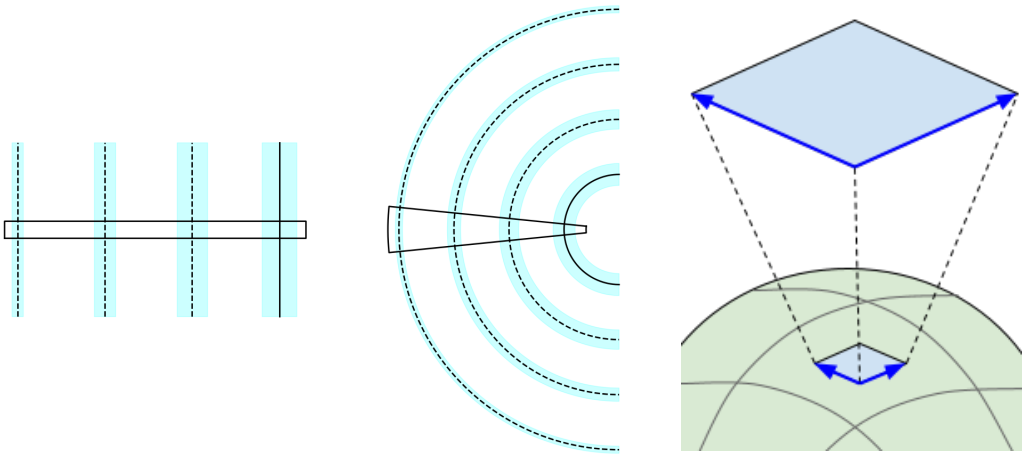


Figure 2.4: *Representation of ill-defined probability density* $\tilde{p}(x) \propto p_Z(f(x))\,\mathrm{vol}\,(f'(\hat{x}))\,e^{-\beta\|\hat{x}-x\|^2}$ (left and center). *Solid black lines denote the manifold, while dashed lines indicate a constant distance from the manifold. The probability density is constant along the manifold. The width of the cyan bands is proportional to* $e^{-\beta\|\hat{x}-x\|^2}$ *and represents the probability density along the on- and off-manifold contours. While the density is reasonable for a flat manifold* (left)*, note that the amount of probability mass associated with a region of the manifold (bounded by solid lines) is* larger *at some points off the manifold than on it when the manifold has curvature* (center)*. This behavior can lead to divergent solutions when optimizing for likelihood and should be compensated for. The appropriate compensation factor is the ratio of the volume of a small region on the manifold* (small blue square embedded in green manifold, right*) to the equivalent region off the manifold* (large blue square, right*). The blue arrows represent an orthonormal frame on the manifold, and the equivalent frame in the off-manifold region.*

We show in the following that we can compensate for the pathological behavior of the on-manifold likelihood loss by evaluating the change of variables term (log-volume term) *off* the manifold, rather than on it.

The (per-sample) loss of the FIF model is (here using the exact maximum likelihood term rather than the surrogate term for the purpose of the derivation):

$$-\log p_Z(f(x)) - \log \operatorname{vol}\left(f'(\hat{x})\right) + \beta\|\hat{x} - x\|^2 \tag{2.131}$$

Consider the probability density $\tilde{p}$ implied by interpreting this loss as a negative log-likelihood:

$$\tilde{p}(x) \propto p_Z(f(x)) \operatorname{vol}\left(f'(\hat{x})\right) e^{-\beta\|\hat{x}-x\|^2} \tag{2.132}$$

Unfortunately, this density is ill-defined and leads to pathological behavior (see Figure 2.4). In order to provide a correction, we need a term that compensates for the volume increase or decrease of off-manifold regions in comparison to the on-manifold region they are projected to. This is depicted in Figure 2.4 (*right*). The blue arrows in the on-manifold region will span the same latent-space volume as the blue arrows in the off-manifold region. The change in volume between the depicted on-manifold region and the latent space is $\operatorname{vol}(f'(\hat{x}))$, and $\operatorname{vol}(f'(x))$ between the off-manifold region and the latent space. Combining these facts means

$$\begin{aligned}\operatorname{vol}(f'(\hat{x})) \times (\text{volume of on-manifold region}) \\ = \operatorname{vol}(f'(x)) \times (\text{volume of off-manifold region})\end{aligned} \tag{2.133}$$

and hence the ratio of the volume of the on-manifold region to the off-manifold region is $\operatorname{vol}(f'(x))/\operatorname{vol}(f'(\hat{x}))$. Multiplying $\tilde{p}(x)$ by this factor leads to

$$\tilde{p}(x)\frac{\operatorname{vol}(f'(x))}{\operatorname{vol}(f'(\hat{x}))} = p_Z(f(x)) \operatorname{vol}\left(f'(x)\right) e^{-\beta\|\hat{x}-x\|^2} \tag{2.134}$$

and the corresponding negative log-likelihood loss is

$$-\log p_Z(f(x)) - \log \operatorname{vol}\left(f'(x)\right) + \beta\|\hat{x} - x\|^2 \tag{2.135}$$

Note what has changed: $f'(x)$ (off-manifold Jacobian matrix) has replaced $f'(\hat{x})$ (on-manifold Jacobian matrix).

Using Lemma 2.2.4, the surrogate for the log-volume term is therefore

$$-\operatorname{tr}(f'(x)\mathtt{SG}[f'(x)^+]) \tag{2.136}$$

In order to maintain computational efficiency, we approximate $f'(x)^+$ by $g'(f(x))$:

$$-\operatorname{tr}(f'(x)\mathtt{SG}[g'(f(x))]) \tag{2.137}$$

which leads to the following off-manifold variant of the maximum likelihood estimator:

$$\mathcal{L}_{\mathrm{ML}}^{\mathrm{off}} = \mathbb{E}_{x,v}\left[-\log p_Z(f(x)) - v^T f'(x)\mathtt{SG}[g'(f(x))v]\right] \tag{2.138}$$

This leads to the following total loss for FIFs:

$$\mathcal{L}_{\mathrm{FIF}} = \mathcal{L}_{\mathrm{ML}}^{\mathrm{off}} + \beta\mathcal{L}_{\mathrm{R}} \tag{2.139}$$

$$= \mathbb{E}_{x,v}\left[-\log p_Z(f(x)) - v^T f'(x)\mathtt{SG}[g'(f(x))v] + \beta\|\hat{x} - x\|^2\right]. \tag{2.140}$$

With this modification, we discourage pathological solutions involving high curvature. In Figure 2.3 *(right)* we can see the effect of the modified estimator: the manifold now moves towards the data since the optimization is not dominated by diverging curvature. We note that the modified estimator is also computationally cheaper than the on-manifold variant, since the vector-Jacobian product $v^T f'(x)$ can reuse the computational graph generated when computing $f(x)$, rather than having to additionally compute $f(\hat{x})$ and a subsequent vector-Jacobian product.

2.2.5 Relationship to rectangular flows

This work is closely related to the previous work on rectangular flows by Caterini et al. (2021). Both approaches involve jointly training a manifold and a distribution on it, using stop-gradient to implement a surrogate loss for the maximum likelihood term. However, they use the on-manifold variant of the loss, which we find to be unstable. Additionally, their implementation is significantly less efficient compared to ours.

A primary difference lies in the relationship used for the surrogate term. They use the expression

$$\nabla \log \mathrm{vol}(J_g) = \nabla \frac{1}{2} \mathbb{E}_v \left[\mathtt{SG} \left[v^T (J_g^T J_g)^{-1} \right] J_g^T J_g v \right] \tag{2.141}$$

where $J_g = g'(f(x))$. This approach results in significantly higher computational costs, mainly because it requires solving an iterative conjugate gradient algorithm to obtain the product with the inverse matrix.

The second major difference is in the use of models. Rectangular flows use injective flows (Brehmer and Cranmer, 2020) (also known as invertible autoencoders (Teng and Choromanska, 2019)), an adaptation to convert normalizing flows into bottleneck models, to jointly parameterize the encoder and decoder. Despite this, a reconstruction loss is still necessary to learn the manifold in the data space. Our strategy is to learn the encoder and decoder separately, without any special architectural constraints. Since they are coupled by a reconstruction loss (which we need anyway), they become approximate functional pseudoinverses during optimization.

Our work improves upon rectangular flows by presenting a significantly more efficient estimator, relaxing architectural constraints, and addressing the curvature-related pathologies that arise from the naive combination of maximum likelihood and manifold (reconstruction loss) optimization.

2.2.6 Implementation details

In implementing the trace estimator, we have to make a number of choices, each elaborated below. For a deeper understanding and more technical details, please refer to Appendix D of Sorrenson et al. (2024b).

Gradient to encoder or decoder

The log-volume term can be formulated either in terms of the Jacobian of the encoder (see Equation (2.110)) or the decoder (see Equation (2.109)). As discussed in Section 2.2.2, we find that formulating it in terms of the encoder Jacobian is conceptually appealing. In practice, it also leads to more stable training. Since the training minimizes the squared

norm of $f(x)$ alongside the log-volume term, we speculate that having gradients from both the squared norm and the log-volume term sent to the encoder allows the encoder to more efficiently shape the latent space distribution. In addition, sending the log-volume gradient to the encoder means the decoder must only minimize reconstruction loss, which implies that it will likely be an approximate pseudoinverse for the encoder, a condition we require for the accuracy of the surrogate estimator.

Space in which trace is performed

	Gradient to encoder	Gradient to decoder
Trace in data space	$-\operatorname{tr}\left(g'(z)\left(\nabla f'(x)\right)\right)$	$\operatorname{tr}\left(\left(\nabla g'(x)\right)f'(x)\right)$
Trace in latent space	$-\operatorname{tr}\left(\left(\nabla f'(x)\right)g'(z)\right)$	$\operatorname{tr}\left(f'(x)\left(\nabla g'(z)\right)\right)$

Table 2.1: *Different possible estimators for the gradient of the log-volume.*

A central component of the maximum likelihood estimator is the trace and the Hutchinson estimator for it. Making use of the cyclic property of the trace, i.e., $\operatorname{tr}(A^TB) = \operatorname{tr}(BA^T)$ for any $A, B \in \mathbb{R}^{D\times d}$, we can choose which expansion of the trace to estimate:

$$\sum_{i=1}^{d}(A^TB)_{ii} = \operatorname{tr}(A^TB) = \operatorname{tr}(BA^T) = \sum_{i=1}^{D}(BA^T)_{ii}. \tag{2.142}$$

The variance of a stochastic trace estimator depends on the noise used but is generally roughly proportional to the squared Frobenius norm of the matrix. Given two matrices $A, B \in \mathbb{R}^{D\times d}$ with $d < D$, it is likely that $\|A^TB\|_F^2 < \|BA^T\|_F^2$, due to the larger number of components in the sum in the latter case.

Transferred to our context: In general, the matrices $f'(x) \in \mathbb{R}^{d\times D}$ and $g'(z) \in \mathbb{R}^{D\times d}$ are rectangular and can be multiplied together in either the $f'(x)g'(z) \in \mathbb{R}^{d\times d}$ order or $g'(z)f'(x) \in \mathbb{R}^{D\times D}$ order. Since $d < D$, the former choice will likely result in lower variance when stochastically estimating the trace.

As a result, when the latent space is smaller than the data space, the preferable estimator is the one that performs the trace in the latent space, meaning that products in the estimator have the order $f'(x)g'(z)$ (see Table 2.1).

Type of gradient

Consider the estimator:

$$\operatorname{tr}\left(\left(\nabla f'(x)\right)g'(z)\right) = \nabla\mathbb{E}_v\left[v^Tf'(x)\texttt{SG}[g'(z)]v\right] \tag{2.143}$$

Ignoring the stop gradient operation for now, this requires computing terms of the form $v^Tf'(x)g'(z)v$. In order to avoid calculating full Jacobian matrices, we can implement the calculation using some combination of vector-Jacobian (VJP) or Jacobian-vector (JVP) products, which are efficient to compute with backward-mode and forward-mode automatic differentiation, respectively. Note that we can use the result from one product as the vector for another VJP or JVP. For example, $v_1 := (v^Tf'(x))^T \in \mathbb{R}^D$ yields a vector, so we can compute $v^Tf'(x)g'(z)v = v_1^Tg'(z)v$ via two vector-Jacobian products.

This gives us three choices: (i) backward mode only (two VJPs), (ii) forward mode only (two JVPs), or (iii) a mix of both (one JVP and one VJP). We opt to use mixed mode.

Trace estimator noise

Trace estimators rely on the identity

$$\mathbb{E}_v[v^T A v] = \mathrm{tr}(A\mathbb{E}_v[vv^T]) = \mathrm{tr}(A) \tag{2.144}$$

meaning that we require only $\mathbb{E}_v[vv^T] = \mathbb{I}$ for the noise variable. The choice of noise distribution affects the variance of the estimator. Among all noise vectors whose entries are sampled independently, Rademacher noise has the lowest variance (Hutchinson, 1989). However, if the entries are sampled from a standard normal distribution and then scaled to have length $\sqrt{d}$, where d is the dimension of v, the variance of the estimator is comparable to Rademacher noise (Girard, 1989). This is because the scaled Gaussian noise vectors are no longer independent, but they cover more directions than Rademacher noise (uniformly covering the hypersphere, rather than at a fixed 2^d points). When using a single Hutchinson sample, we choose to use scaled Gaussian noise for its low variance and its ability to cover more directions. When we have more than one Hutchinson sample, we additionally orthogonalize the vectors as this further reduces variance.

Number of noise samples

We can choose to use between 1 and d noise samples to approximate the expectation in the trace estimator (with d samples we can calculate the exact trace, so more samples are not necessary). Denote the number of samples by K. We find that, in general, $K = 1$ is enough for good performance, especially if the batch size is sufficiently high.

2.3 Manifold free-form flows

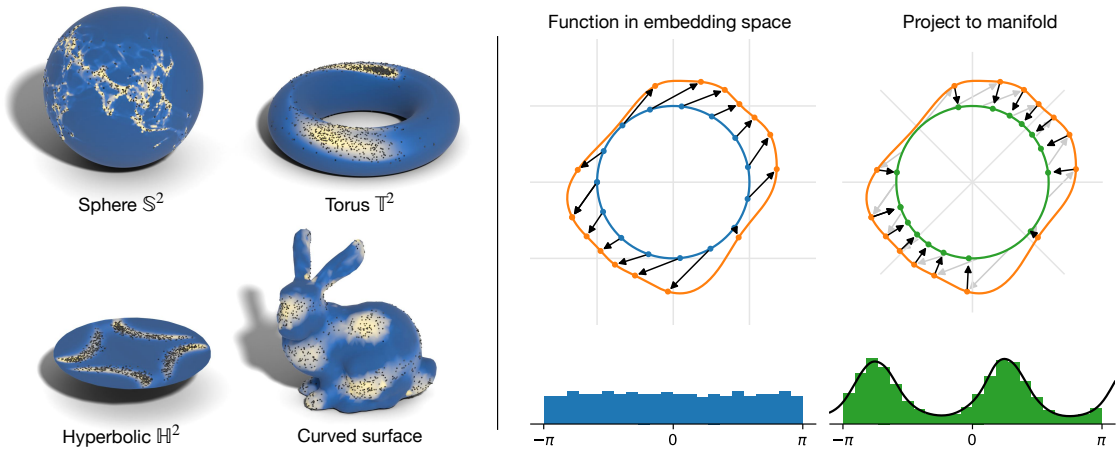


Figure 2.5: *Manifold free-form Flows (M-FFF) learn generative models on a variety of manifolds.* Left: *The learned distributions* (colored surface) *accurately match the test points* (black dots). Right: *We parameterize M-FFF using a neural network in an embedding space, whose outputs are projected onto the manifold. This enables simulation-free training and inference, and naturally respects the corresponding geometry, yielding fast sampling and continuous distributions regardless of the manifold.*

This section is based on work previously published as "Learning distributions on manifolds with free-form flows" at NeurIPS 2024 (Sorrenson et al., 2024a). All figures are reproduced from that publication.

	Respects topology	Single-step sampling	Arbitrary manifolds
Euclidean	✗	✓	✓
Specialized	✓	✓	✗
Continuous time	✓	✗	✓
M-FFF (ours)	✓	✓	✓

Table 2.2: *Feature comparison of generative models on manifolds.*

The free-form flow framework naturally extends to Riemannian manifolds, allowing us to define a new class of generative models on such manifolds. The resulting manifold free-form flows (M-FFFs) have distinct advantages over existing approaches (see Table 2.2). Briefly, existing approaches fall into three categories:

1. **Euclidean generative models:** These models reduce the problem to learning a distribution in Euclidean space (Gemici et al., 2016; Falorsi et al., 2019; Kalatzis et al., 2021; Brofos et al., 2021). A downside is that they may not respect topology; for example, mapping a sphere to a plane can cause discontinuities.

2. **Specialized architectures:** These involve coupling blocks for normalizing flows designed to work on specific manifolds, such as $SO(3)$ (Liu et al., 2023), $SU(d)$ and $U(d)$ (Boyda et al., 2021; Kanwar et al., 2020), hyperbolic space (Bose et al., 2020), as well as tori and spheres (Rezende et al., 2020).

3. **Continuous-time models:** These models parameterize an ODE or SDE in the tangent space of a Riemannian manifold (Rozen et al., 2021; Falorsi, 2021; Falorsi

and Forré, 2020; Huang et al., 2022; Mathieu and Nickel, 2020; De Bortoli et al., 2022; Chen and Lipman, 2024; Lou et al., 2020; Ben-Hamu et al., 2022). Sampling and evaluating densities require iterative solving of the differential equation.

2.3.1 Free-form manifold-to-manifold neural networks

In analogy to free-form flows defined in Euclidean space, we learn a pair of free-form neural networks: an encoder $f(x)$ and a decoder $g(z)$. To respect the geometry of the setting, these are implemented as manifold-to-manifold functions:

$$f(x) : \mathcal{M} \to \mathcal{M}, \quad g(z) : \mathcal{M} \to \mathcal{M} \tag{2.145}$$

We implement such functions using feed-forward neural networks $\tilde{f}, \tilde{g} : \mathbb{R}^m \to \mathbb{R}^m$ that operate in an embedding space $\mathbb{R}^m$ of $\mathcal{M}$, ensuring that their outputs lie on the manifold by appending a projection $\pi : \mathbb{R}^m \to \mathcal{M}$, which maps points from the embedding space $\mathbb{R}^m$ to the manifold $\mathcal{M}$:

$$f(x) = \pi(\tilde{f}(x)), \quad g(z) = \pi(\tilde{g}(z)) \tag{2.146}$$

Figure 2.5 illustrates this for an example on a circle $\mathcal{M} = \mathbb{S}^1$.

Similar to the Euclidean case, we use a reconstruction loss to ensure that f and g are approximately inverses of each other:

$$\mathcal{L}_{\mathrm{R}} = \mathbb{E}_x[\|g(f(x)) - x\|^2] \tag{2.147}$$

We calculate the distance in the embedding space; an alternative approach would be to use a distance metric specific to the manifold (e.g., the great circle distance for the sphere), but we find that the ambient Euclidean distance is effective in practice, as it closely approximates the manifold distance for small distances.

Our loss function is very similar to that of the Euclidean free-form flow:

$$\mathcal{L}_{\text{M-FFF}} = \mathbb{E}_{x,v}\left[-\log p_Z(f(x)) - v^T f'(x)\mathtt{SG}[g'(f(x))v]\right] + \beta\mathcal{L}_{\mathrm{R}} \tag{2.148}$$

We will show how to adapt FFF to Riemannian manifolds in the following sections.

2.3.2 Manifold change of variables

In this section, we will focus on intuitive definitions of concepts from topology and differential geometry. For a more rigorous treatment, see Jost (2008).

An n-dimensional manifold $\mathcal{M}$ is a space where every point x has a neighborhood that is homeomorphic to an open subset of $\mathbb{R}^n$. Intuitively, this means that there is a small region of $\mathcal{M}$ containing x that can be bent and stretched in a continuous way to map onto a small region in $\mathbb{R}^n$. This is what is meant when we say that the manifold locally resembles $\mathbb{R}^n$. If all these maps from $\mathcal{M}$ to $\mathbb{R}^n$ are also differentiable, then the manifold itself is differentiable, as long as there is a way to connect the local neighborhoods in a differentiable and consistent way.

The tangent space of the manifold at x, denoted $\mathcal{T}_x\mathcal{M}$, is an n-dimensional Euclidean space, which is a linearization of the manifold at x: if we zoom in to a very small

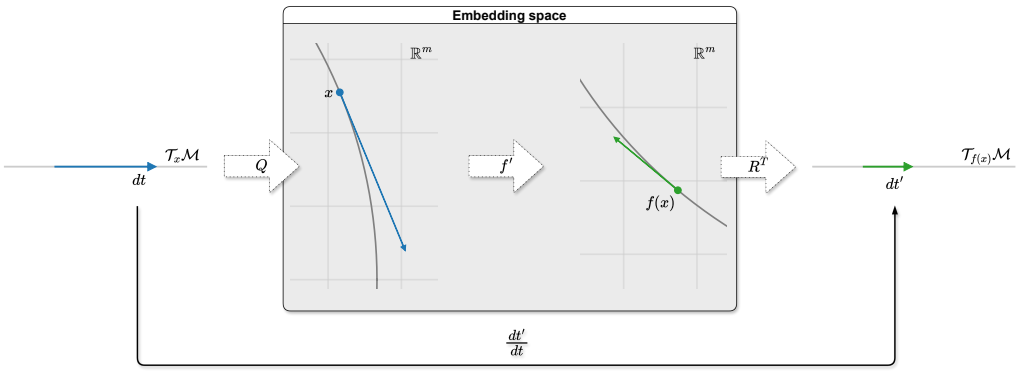


Figure 2.6: *Computation of the volume change in the tangent space of the manifold: The manifold change of variables formula in Equation* (2.156) *requires computing the change of a volume element in the tangent spaces under f, which in this example is given by the ratio of lengths of dt and dt'. Since f is a map in the embedding space, $f'(x)$ defines a linear map between vectors from the embedding space. To correctly compute the change in volume, we use Q and R to change coordinates to the intrinsic tangent spaces, resulting in the linear map $R^T f'(x)Q : \mathcal{T}_x\mathcal{M} \to \mathcal{T}_{f(x)}\mathcal{M}$, which maps dt to dt'.*

region around x, the manifold looks flat, and this flat Euclidean space is aligned with the tangent space. Because the tangent space is a linearization of the manifold, this is where derivatives on the manifold live; e.g., if $f : \mathcal{M}_X \to \mathcal{M}_Z$ is a map between two manifolds, then the Jacobian $f'(x)$ is a linear map from $\mathcal{T}_x\mathcal{M}_X$ to $\mathcal{T}_{f(x)}\mathcal{M}_Z$.

A Riemannian manifold $(\mathcal{M}, G)$ is a differentiable manifold that is equipped with a Riemannian metric $G : \mathcal{T}_x\mathcal{M} \times \mathcal{T}_x\mathcal{M} \to \mathbb{R}$, which defines an inner product on the tangent space, allowing us to calculate lengths and angles in this space. The length of a smooth curve $\gamma : [0, 1] \to \mathcal{M}$ is given by the integral of the length of its velocity vector $\gamma'(t) \in \mathcal{T}_{\gamma(t)}\mathcal{M}$. This ultimately allows us to define a notion of distance on the manifold, as the curve of minimal length connecting two points.

In the following, we always consider Riemannian manifolds.

Embedded manifolds

We define an embedded manifold and its properties as follows:

Definition 2.3.1 (Embedded Manifold)**.** *An n-dimensional manifold $\mathcal{M}$ embedded in $\mathbb{R}^m$ is defined via a projection function*

$$\pi : \mathbb{P} \to \mathbb{R}^m \tag{2.149}$$

where $\mathbb{P} \subseteq \mathbb{R}^m$ is the projectable set. The projection π must satisfy:

1. *$\pi \circ \pi = \pi$.*
2. *π is smooth on $\mathbb{P}$.*
3. *$\operatorname{rank}(\pi'(\pi(x))) = n$ for all $x \in \mathbb{P}$.*

The manifold $\mathcal{M}$ is then defined as the set that projects onto itself:

$$\mathcal{M} = \{x \in \mathbb{R}^m : \pi(x) = x\} \tag{2.150}$$

The tangent space of $\mathcal{M}$ at a point x is defined as the column space of the Jacobian

matrix of π:

$$\mathcal{T}_x\mathcal{M} = \operatorname{col}(\pi'(x)) \tag{2.151}$$

An example of an embedded manifold is the unit sphere $\mathbb{S}^{m-1}$ in $\mathbb{R}^m$. The projection function for the sphere is given by:

$$\pi(x) = \frac{x}{\|x\|} \tag{2.152}$$

where $\|x\|$ is the Euclidean norm of x. The Jacobian of the projection for the unit sphere can be derived as follows:

$$\pi'(x) = \frac{1}{\|x\|}\mathbb{I} - \frac{1}{\|x\|^3}xx^T \tag{2.153}$$

where $\mathbb{I}$ is the $m \times m$ identity matrix.

This projection satisfies the properties of an embedded manifold:

1. $\pi \circ \pi = \pi$: Projecting a point that is already on the sphere does not change its position.
2. π is smooth on $\mathbb{P} = \mathbb{R}^m \setminus \{0\}$: The projection is differentiable everywhere except at the origin.
3. $\pi'(x)$ is a full-rank matrix minus a rank-one matrix. A matrix with this structure could have either rank m or $m-1$. Since $\pi'(x)x = 0$, it cannot be full rank, so the rank of $\pi'(x)$ is $m-1$ for all $x \in \mathbb{S}^{m-1}$.

Since $\pi'(x)x = 0$, the tangent space is the hyperplane orthogonal to x:

$$\mathcal{T}_x\mathbb{S}^{m-1} = \{v \in \mathbb{R}^m : v^T x = 0\} \tag{2.154}$$

Integration on embedded manifolds

In order to perform integration on the manifold, we cannot work directly in the m-dimensional coordinates of the embedding space; instead, we have to introduce some local n-dimensional coordinates. This means that the domain of integration has to be diffeomorphic to an open set in $\mathbb{R}^n$. Since this might not be the case for the whole region of integration, we might need to partition it into such regions and perform integration on each individually (each such region, together with its map to $\mathbb{R}^n$, is known as a chart, and a collection of charts is an atlas). For example, if we want to integrate a function on the sphere, we could split the sphere into two hemispheres and integrate each separately. A hemisphere can be continuously stretched and flattened into a two-dimensional region, whereas the whole sphere cannot without creating discontinuities.

Given an open set U in $\mathbb{R}^n$, and a diffeomorphic local embedding function $\phi : U \to \mathcal{M}$, the integral of a scalar function $p : \mathcal{M} \to \mathbb{R}$ on $\phi(U) \subseteq \mathcal{M}$ is

$$\int_{\phi(U)} p \,\mathrm{d}V = \int_U (p \circ \phi)\sqrt{|\phi'(u)^T G(\phi(u))\phi'(u)|}\,\mathrm{d}u^1 \cdots \mathrm{d}u^n \tag{2.155}$$

The integral on the right is an ordinary integral in $\mathbb{R}^n$. The quantity inside the determinant is known as the pullback metric.

Using this formalism, we can prove the following theorem, which generalizes the change of variables formula to Riemannian manifolds.

Theorem 2.3.1 (Manifold change of variables). *Let $(\mathcal{M}_X, G_X)$ and $(\mathcal{M}_Z, G_Z)$ be n-dimensional Riemannian manifolds embedded in $\mathbb{R}^m$, i.e., $\mathcal{M}_X, \mathcal{M}_Z \subseteq \mathbb{R}^m$, and assume they have the same global topological structure. Let p_X be a probability distribution on $\mathcal{M}_X$ and let $f : \mathcal{M}_X \to \mathcal{M}_Z$ be a diffeomorphism. Let p_Z be the pushforward of p_X under f.*
Let $x \in \mathcal{M}_X$. Define $Q \in \mathbb{R}^{m \times n}$ as an orthonormal basis for $\mathcal{T}_x \mathcal{M}_X$ and $R \in \mathbb{R}^{m \times n}$ as an orthonormal basis for $\mathcal{T}_{f(x)} \mathcal{M}_Z$.
Then, the probability densities p_X and p_Z are related under the change of variables $x \mapsto f(x)$ by the following equation:

$$p_X(x) = p_Z(f(x)) \cdot |R^T f'(x) Q| \cdot \sqrt{\frac{|R^T G_Z(f(x)) R|}{|Q^T G_X(x) Q|}} \tag{2.156}$$

where Q and R depend on x and $f(x)$, respectively, although this dependency is omitted for brevity.

Proof. Let $\phi : \mathbb{R}^n \to \mathcal{M}_X$ be defined by $\phi(u) = \pi_X(x + Qu)$. Let U be an open subset of $\mathbb{R}^n$ containing the origin that is small enough so that ϕ is bijective. Let $\psi : \mathbb{R}^n \to \mathcal{M}_Z$ be defined by $\psi(w) = \pi_Z(f(x) + Rw)$. Define $\varphi = \psi^{-1} \circ f \circ \phi$ and let $W = \varphi(U)$.
Note that $\phi'(u) = \pi_X'(x + Qu) \cdot Q$ and hence $\phi'(0) = \pi_X'(x) Q = Q$ (since each column of Q is in $\mathcal{T}_x \mathcal{M}_X = \operatorname{col}(\pi_X'(x))$).
Similarly, $\psi'(0) = R$. Since ψ is a map from n to m dimensions, there is not a unique function from $\mathbb{R}^m$ to $\mathbb{R}^n$ that is ψ^{-1} on the manifold, and there are remaining degrees of freedom in the off-manifold behavior that can result in different Jacobians. For our purposes, we define the inverse ψ^{-1} such that $\psi \circ \psi^{-1}$ is an orthogonal projection onto $\mathcal{M}_Z$. This means $\psi'(\psi^{-1}(f(x)))(\psi^{-1})'(f(x)) = RR^T$ and hence $(\psi^{-1})'(f(x)) = R^T$.
Since p_Z is the pushforward of p_X under f, the amount of probability mass contained in $\phi(U)$ is the same as that contained in $f(\phi(U)) = \psi(W)$:

$$\int_{\phi(U)} p_X(x) \, \mathrm{d}V_X = \int_{\psi(W)} p_Z(z) \, \mathrm{d}V_Z \tag{2.157}$$

and therefore:

$$\begin{aligned} &\int_U p_X(\phi(u)) \sqrt{|\phi'(u)^T G_X(\phi(u)) \phi'(u)|} \, \mathrm{d}u^1 \cdots \mathrm{d}u^n \\ &\qquad = \int_W p_Z(\psi(w)) \sqrt{|\psi'(w)^T G_Z(\psi(w)) \psi'(w)|} \, \mathrm{d}w^1 \cdots \mathrm{d}w^n \end{aligned} \tag{2.158}$$

Changing variables of the RHS with $w = \varphi(u)$ gives us

$$\begin{aligned} &\int_U p_X(\phi(u)) \sqrt{|\phi'(u)^T G_X(\phi(u)) \phi'(u)|} \, \mathrm{d}u^1 \cdots \mathrm{d}u^n \\ &\quad = \int_U p_Z(f(\phi(u))) \sqrt{|\psi'(\varphi(u))^T G_Z(f(\phi(u))) \psi'(\varphi(u))|} \cdot \left| \frac{\partial w}{\partial u} \right| \mathrm{d}u^1 \cdots \mathrm{d}u^n \end{aligned} \tag{2.159}$$

Since U was arbitrary, we can make it arbitrarily small, demonstrating that the integrands must be equal for $u = 0$:

$$p_X(x)\sqrt{|Q^T G_X(x)Q|} = p_Z(f(x))\sqrt{|R^T G_Z(f(x))R|} \cdot \left|\frac{\partial w}{\partial u}\right| \tag{2.160}$$

Since $w = \psi^{-1}(f(\phi(u)))$, the Jacobian has the following form when evaluated at the origin (note $\phi(0) = x$):

$$\left.\frac{\partial w}{\partial u}\right|_{u=0} = (\psi^{-1})'(f(x)) \cdot f'(x) \cdot \phi'(0) \tag{2.161}$$

$$= R^T f'(x) Q \tag{2.162}$$

Substituting this into the equality and rearranging gives the result:

$$p_X(x) = p_Z(f(x)) \cdot |R^T f'(x)Q| \cdot \sqrt{\frac{|R^T G_Z(f(x))R|}{|Q^T G_X(x)Q|}} \tag{2.163}$$

□

2.3.3 Loss function

This theorem extends the Euclidean free-form flow approximation of the log-Jacobian to Riemannian manifolds. The result closely mirrors the Euclidean case, with one key difference: the random vector v must have a different covariance matrix, which in practice means it should be sampled from the tangent space at $f(x)$.

Theorem 2.3.2. *Under the assumptions of Theorem 2.3.1, let $v \in \mathbb{R}^m$ be a random variable with zero mean and covariance RR^T. Denote $g = f^{-1}$. Then, the derivative of the change of variables term has the following trace expression:*

$$\nabla \log |R^T f'(x)Q| = \operatorname{tr}(R^T(\nabla f'(x))g'(f(x))R) \tag{2.164}$$

$$= \mathbb{E}_v\left[v^T(\nabla f'(x))g'(f(x))v\right]. \tag{2.165}$$

Proof. First, a reminder that $\varphi'(0) = R^T f'(x)Q$ with $\varphi = \psi^{-1} \circ f \circ \phi$. Let $\chi = \varphi^{-1}$, i.e., $\chi = \phi^{-1} \circ g \circ \psi$. Jacobi's formula tells us that

$$\frac{\mathrm{d}}{\mathrm{d}t}\log|A(t)| = \operatorname{tr}\left(\frac{\mathrm{d}A(t)}{\mathrm{d}t}A(t)^{-1}\right) \tag{2.166}$$

Note also that since $\chi(\varphi(u)) = u$, therefore $\chi'(\varphi(u))\varphi'(u) = \mathbb{I}$ and $\chi'(\varphi(u)) = \varphi'(u)^{-1}$. Applying Jacobi's formula to $\varphi'(0)$:

$$\nabla \log|\varphi'(0)| = \operatorname{tr}((\nabla\varphi'(0))\varphi'(0)^{-1}) \tag{2.167}$$

$$= \operatorname{tr}((\nabla\varphi'(0))\chi'(\varphi(0))) \tag{2.168}$$

and substituting in f and g:

$$\nabla \log|R^T f'(x)Q| = \operatorname{tr}(\nabla(R^T f'(x)Q)Q^T g'(f(x))R) \tag{2.169}$$

Q does not depend on the derivative, but R depends on $f(x)$ and hence must be considered in the derivative. However,

$$\nabla \operatorname{tr}(RR^T) = \operatorname{tr}((\nabla R)R^T + R\nabla R^T) = 2\operatorname{tr}(R\nabla R^T) \tag{2.170}$$

and since $\operatorname{tr}(RR^T) = \operatorname{tr}(\mathbb{I})$ is a constant, we have $\operatorname{tr}(R\nabla R^T) = 0$. Expanding Equation (2.169):

$$\begin{aligned}\nabla \log|R^T f'(x)Q| = \operatorname{tr}(\nabla(R^T)f'(x)QQ^T g'(f(x))R) \\ + \operatorname{tr}(R^T\nabla(f'(x))QQ^T g'(f(x))R)\end{aligned} \tag{2.171}$$

Since Q is an orthonormal basis for $\mathcal{T}_x\mathcal{M}_X$, QQ^T is a projection matrix onto $\mathcal{T}_x\mathcal{M}_X$. This is because $(QQ^T)^2 = QQ^TQQ^T = QQ^T$. As a result, $QQ^T\pi'(x) = \pi'(x)$. Since g can also be written inside a projection: $g(z) = \pi_Z(g(z))$, therefore $g'(z) = \pi'_Z(\tilde{g}(z))\tilde{g}'(z)$, so $QQ^Tg'(z) = g'(z)$. Note also that $f'(x)g'(f(x)) = \mathbb{I}$ since $f \circ g = \mathrm{id}$. This simplifies the equation:

$$\nabla \log|R^T f'(x)Q| = \operatorname{tr}(\nabla(R^T)R) + \operatorname{tr}(R^T\nabla(f'(x))g'(f(x))R) \tag{2.172}$$

and finally

$$\nabla \log|R^T f'(x)Q| = \operatorname{tr}(R^T\nabla(f'(x))g'(f(x))R) \tag{2.173}$$

Finally, for a square matrix A we have

$$\mathbb{E}_v[v^TAv] = \operatorname{tr}(A\mathbb{E}_v[vv^T]) = \operatorname{tr}(ARR^T) = \operatorname{tr}(R^TAR) \tag{2.174}$$

meaning

$$\operatorname{tr}(R^T(\nabla f'(x))g'(f(x))R) = \mathbb{E}_v\left[v^T(\nabla f'(x))g'(f(x))v\right] \tag{2.175}$$

□

In the above proof, we used the fact that $QQ^Tg'(z) = g'(z)$. Can we use $RR^Tf'(x) = f'(x)$ to simplify the equation further? No, we cannot, since the expression involving f' is actually its derivative with respect to parameters, which may not have the same matrix structure as f'. Is it instead possible to use $g'(z)RR^T = g'(z)$ for simplification? If g is implemented as $\pi_Z(\tilde{g}(z))$, this is not necessarily true, as $g'(z)$ might not be a map from the tangent space at z to the tangent space at $g(z)$. For example, if we add a small deviation v to z, where v is orthogonal to the tangent space at z, then $g(z+v)$ might not equal $g(z)$. However, this would mean that derivatives in the off-manifold direction can be non-zero, meaning that $g'(z)v \neq g'(z)RR^Tv = 0$ (since RR^T will project v to 0). We can change this by prepending g with a projection:

$$g = \pi_X \circ \tilde{g} \circ \pi_Z. \tag{2.176}$$

If π_Z is an orthogonal projection, meaning that π'_Z is symmetric, the column space and row space of π_Z will both be the same as those of RR^T, meaning $\pi'_Z(z)RR^T = \pi'_Z$ and hence $g'(z)RR^T = g'(z)$. This leads to the following corollary:

Corollary 2.3.2.1. *Suppose the assumptions of Theorem 2.3.1 hold with the following implementation for $g = f^{-1}$*

$$g = \pi_X \circ \tilde{g} \circ \pi_Z \tag{2.177}$$

where π_Z is an orthogonal projection. Then the derivative of the change of variables term has the following trace expression:

$$\nabla \log |R^T f'(x) Q| = \operatorname{tr}((\nabla f'(x)) g'(f(x))). \tag{2.178}$$

Proof. Take the result of Theorem 2.3.2 and use the cyclic property of the trace and the properties of g' discussed above:

$$\begin{aligned} \operatorname{tr}(R^T \nabla(f'(x)) g'(f(x)) R) &= \operatorname{tr}(\nabla(f'(x)) g'(f(x)) R R^T) && (2.179) \\ &= \operatorname{tr}(\nabla(f'(x)) g'(f(x))). && (2.180) \end{aligned}$$

□

We have two variants of the trace estimate derived above, one evaluated in $\mathbb{R}^n$ (Theorem 2.3.2), the other in $\mathbb{R}^m$ (Corollary 2.3.2.1). The first can be estimated using the following equality:

$$\begin{aligned} \operatorname{tr}(R^T \nabla(f'(x)) g'(f(x)) R) &= \mathbb{E}_u[u^T R^T \nabla(f'(x)) g'(f(x)) R u] && (2.181) \\ &= \mathbb{E}_v[v^T \nabla(f'(x)) g'(f(x)) v] && (2.182) \\ &= \nabla \mathbb{E}_v[v^T f'(x) \mathtt{SG}[g'(f(x))] v] && (2.183) \end{aligned}$$

where $\mathbb{E}_u[uu^T] = \mathbb{I}_n$ for $u \in \mathbb{R}^n$ and $p(v)$ is the distribution of Ru, which lies in the tangent space at x with $\mathbb{E}[vv^T] = RR^T$. An example of such a distribution is the standard normal projected to the tangent space, which we can achieve by $v = \pi'(x)\tilde{v}$ where $\tilde{v}$ is standard normal.

In the second case, we can just sample from a distribution where $\mathbb{E}_v[vv^T] = \mathbb{I}_m$ in the embedding space $\mathbb{R}^m$:

$$\operatorname{tr}(\nabla(f'(x)) g'(f(x))) = \nabla \mathbb{E}_v[v^T \nabla(f'(x)) \mathtt{SG}[g'(f(x))] v]. \tag{2.184}$$

Variance reduction

When using a Hutchinson trace estimator with standard normal $v \in \mathbb{R}^n$, we can reduce the variance of the estimate by scaling v to have length $\sqrt{n}$ (see Girard (1989)). The scaled variable will still have zero mean and unit covariance, ensuring the estimate remains unbiased. However, the variance is reduced, with the effect especially pronounced in low dimensions.

While we can take advantage of this effect in both our options for trace estimator, the effect is more pronounced in lower dimensions, so we reduce the variance more by estimating the trace in an n-dimensional space rather than an m-dimensional space. Hence the first version of the trace estimator, where v is sampled from a distribution in $\mathcal{T}_x \mathcal{M}_X$, is preferable in this regard.

Let's provide some intuition with an example. Suppose $n = 1$, $m = 2$ and $R = (1, 0)^T$. We want to estimate the trace of $A = \mathrm{diag}(1, 0)$. Using the first estimator, we first sample $v = RR^T\tilde{v}$ with $\tilde{v}$ standard normal which results in $v = (u, 0)^T$ where $u \in \mathbb{R}$ is standard normal. Then we scale v so it has length $\sqrt{n} = 1$. This results in $v = (r, 0)^T$ where r is a Rademacher variable (taking the values -1 and 1 with equal probability). The trace estimate is therefore $r^2 = 1$, meaning we always get the correct answer, so the variance is zero. The second estimator samples v directly from a 2D standard normal, then scales it to have length $\sqrt{m} = \sqrt{2}$. Hence v is sampled uniformly from the circle with radius $\sqrt{2}$. We can write $v = \sqrt{2}(\cos\theta, \sin\theta)^T$ with θ sampled uniformly in $[0, 2\pi]$. The estimate $v^T Av = 2\cos^2\theta$. This is a random variable whose mean is indeed 1 as required but has a nonzero variance, showing that the variance is higher when estimating in the m-dimensional space.

For this reason, we choose the first estimator, sampling v in the tangent space at x. This also simplifies the definition of g, meaning that we don't have to prepend it with a projection.

2.4 Experiments

All figures and tables in this section are taken from the relevant papers:

1. Free-form flows (full-dimensional) (Draxler et al., 2024)
2. Free-form injective flows (Sorrenson et al., 2024b)
3. Manifold free-form flows (Sorrenson et al., 2024a)

2.4.1 Free-form flows (full-dimensional)

Effect of reconstruction loss weight on invertibility

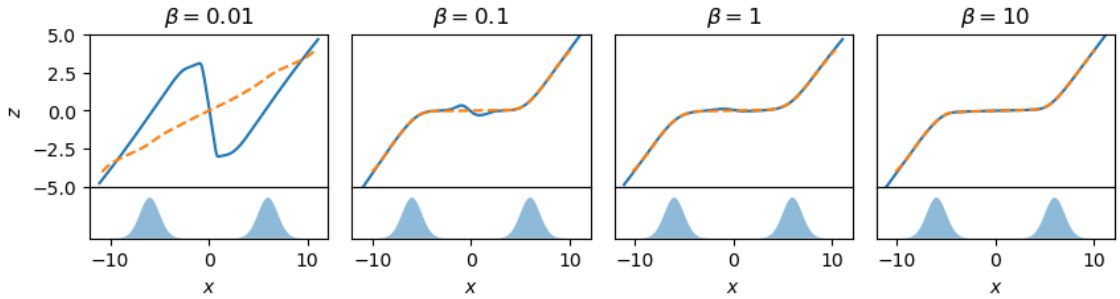


Figure 2.7: *Learned solutions to $\mathcal{L}_{\mathrm{NF}} + \beta\mathcal{L}_{\mathrm{R}}$ for various β. The data is the two-component Gaussian mixture shown in the lower panels. Solid blue lines show f and dashed orange lines show g. Note that f is not globally invertible when β is small.*

Figure 2.7 demonstrates the behavior of learning $\mathcal{L}_{\mathrm{NF}} + \beta\mathcal{L}_{\mathrm{R}}$ without restricting f to diffeomorphisms. The behavior is as predicted in Theorem 2.1.8. For low values of β, we observe that the learned solution is not invertible between the data modes. In the extreme case (leftmost panel), the function learns to map each mixture component independently to the latent standard normal distribution. This approach yields a high likelihood but poor reconstruction quality.

As we increase β, and consequently the reconstruction loss, we observe a progression towards more globally consistent solutions. The function gradually learns to maintain invertibility across the entire domain, not just within each mode. This progression culminates in a globally invertible solution for sufficiently large β (rightmost panel), which closely resembles the behavior of traditional normalizing flows.

This experiment highlights the trade-off between likelihood and reconstruction quality, and demonstrates the importance of choosing a sufficiently high β value when optimizing free-form networks with the normalizing flow loss.

Simulation-based inference

Simulation-based inference (SBI) involves estimating the parameters of a model or simulation from observations. As multiple parameter sets can lead to similar observations, a generative model is necessary, allowing us to sample from a distribution over parameters. We train on simulations to learn a conditional generative model that predicts parameters (inputs) from outputs.

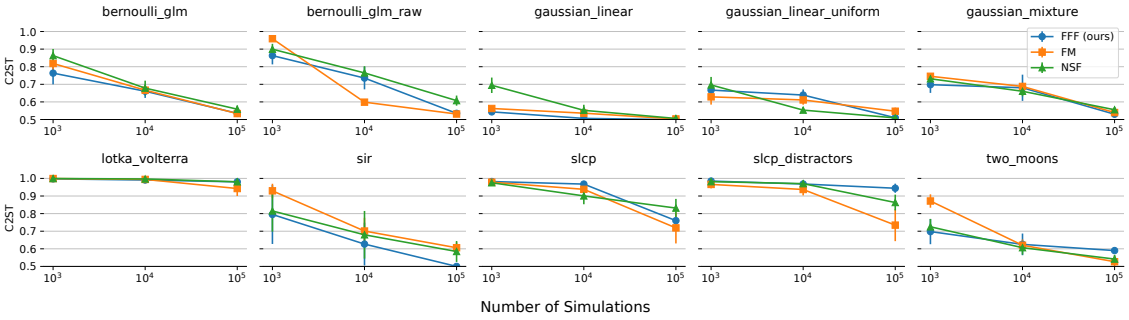


Figure 2.8: *C2ST scores (lower is better) on the SBI benchmark datasets. We compare our method (FFF) against flow matching (FM) (Wildberger et al., 2023) and the neural spline flow (NSF) baseline in the benchmark dataset (Lueckmann et al., 2021). The scores are averaged over ten different observations, with error bars indicating the standard deviation. Our performance is comparable to the competitors across all datasets, with no model being universally better or worse.*

We use the benchmark from Lueckmann et al. (2021), which comprises 10 inverse problems of varying difficulty, each with three simulation budgets (number of training examples). Performance is evaluated using the classifier 2-sample test (C2ST) (Lopez-Paz and Oquab, 2017; Friedman, 2003), where the score is determined by a classifier trained to distinguish between generated and true posterior samples. Scores range between 0.5 (best) and 1 (worst).

Figure 2.8 compares the C2ST scores of our model against SBI baselines: neural spline flows (Durkan et al., 2019) and flow matching (Wildberger et al., 2023). Our method demonstrates competitive performance, particularly at low budgets, and is straightforward to tune. We employ nearly identical architecture across all tasks, only widening networks for larger datasets. For complete experimental details, refer to Appendix C.1 of Draxler et al. (2024).

Molecule generation

We evaluate our method on two molecule generation benchmarks: as a Boltzmann generator on three synthetic molecule-like energy distributions, and as a generator of molecules from the QM9 dataset.

In both cases, we use a network equivariant to rotations and translations ($E(3)$ equivariant), utilizing the $E(n)$-GNN proposed by Satorras et al. (2021b). This approach respects the symmetry of molecules and leads to faster learning. Our latent distribution is invariant to $E(3)$ transformations, ensuring that the overall learned density is also invariant. See Appendix C.2 in Draxler et al. (2024) for full experimental details.

Boltzmann generators A Boltzmann generator (Noé et al., 2019) is a generative model for a distribution defined by a known energy function $u(x)$:

$$p(x) \propto e^{-u(x)} \tag{2.185}$$

Since we know $u(x)$, we can reweight samples from our learned distribution $q(x)$ to obtain unbiased estimates of expectations of observables:

$$\mathbb{E}_{p(x)}[O(x)] = \mathbb{E}_{q(x)}\left[O(x)\frac{p(x)}{q(x)}\right] = \frac{\mathbb{E}_{q(x)}[O(x)e^{-u(x)-\log q(x)}]}{\mathbb{E}_{q(x)}[e^{-u(x)-\log q(x)}]} \tag{2.186}$$

	NLL (↓)	Sampling time (↓)
	DW4	
$E(n)$-NF (Satorras et al., 2021a)	1.72 ± 0.01	**0.024** ms
OT-FM (Klein et al., 2023)	**1.70** ± 0.02	0.034 ms
E-OT-FM (Klein et al., 2023)	**1.68** ± 0.01	0.033 ms
$E(n)$-FFF (*ours*)	**1.68** ± 0.01	**0.026** ms
	LJ13	
$E(n)$-NF	-16.28 ± 0.04	0.27 ms
OT-FM	-16.54 ± 0.03	0.77 ms
E-OT-FM	-16.70 ± 0.12	0.72 ms
$E(n)$-FFF (*ours*)	**-17.09** ± 0.16	**0.11** ms
	LJ55	
OT-FM	-88.45 ± 0.04	40 ms
E-OT-FM	**-89.27** ± 0.04	40 ms
$E(n)$-FFF (*ours*)	-88.72 ± 0.16	**2.1** ms

Table 2.3: *Equivariant free-form flows ($E(n)$-FFF) sample significantly faster than previous models and achieve comparable or better negative log-likelihood (NLL, lower is better). Best results in bold.*

	NLL (↓)	Stable (↑)	Sampling time (↓)	
			Raw	Stable
$E(3)$-NF	-59.7	4.9 %	13.9 ms	309.5 ms
$E(3)$-DM	**-110.7**	**82.0** %	1580.8 ms	1970.6 ms
$E(3)$-FFF	-76.2	8.7 %	**0.6** ms	**8.1** ms
Data	-	95.2 %	-	-

Table 2.4: *$E(3)$-FFF (ours) trained on QM9 generates stable molecules faster than previous models because a sample is obtained via a single function evaluation. $E(3)$-DM is the $E(3)$-diffusion model (Hoogeboom et al., 2022), and $E(3)$-NF is the $E(3)$-normalizing flow (Satorras et al., 2021a). The latter is also trained explicitly using maximum likelihood, yet is outperformed by $E(3)$-FFF in terms of negative log-likelihood (NLL, lower is better) and the fraction of generated molecules that are stable. Best results in bold.*

We evaluate on three benchmark tasks: DW4, LJ13, and LJ55 (Köhler et al., 2020; Klein et al., 2023). Each task involves summing up pairwise potentials between particles:

- DW4: Uses a double well potential with 4 particles in 2D.
- LJ13: Uses the Lennard-Jones potential with 13 particles in 3D.
- LJ55: Uses the Lennard-Jones potential with 55 particles in 3D.

Samples for these tasks come from MCMC and are provided by Klein et al. (2023).

In Table 2.3, we compare our method against a continuous $E(3)$ equivariant normalizing flow (neural ODE) (Satorras et al., 2021a) and two equivariant ODEs trained with optimal transport (OT) flow matching (Klein et al., 2023). Our method achieves comparable or better negative log-likelihood (NLL) and is significantly faster, especially as the data complexity increases (e.g., LJ55).

QM9 molecules The QM9 dataset (Ruddigkeit et al., 2012; Ramakrishnan et al., 2014) contains molecules with varying atom counts, up to 29 atoms per molecule. Each atom is represented by its 3D coordinates and additional properties such as atom type and charge.

As a single-step model, FFF is significantly faster than continuous models like diffusion or flow matching. As shown in Table 2.4, our method can generate stable molecules two orders of magnitude faster than the $E(3)$ diffusion model (Hoogeboom et al., 2022) and one order of magnitude faster than the $E(3)$ normalizing flow (Satorras et al., 2021a).

Compared to the $E(3)$ normalizing flow, FFF achieves better negative log-likelihood and generates a higher percentage of stable molecules. However, it's worth noting that the $E(3)$ diffusion model still outperforms FFF in terms of both the number of stable molecules generated and negative log-likelihood. This trade-off between speed and performance highlights the potential of FFF as a fast, high-quality alternative in scenarios where rapid generation is crucial.

2.4.2 Free-form injective flows

Trade-off between likelihood and reconstruction

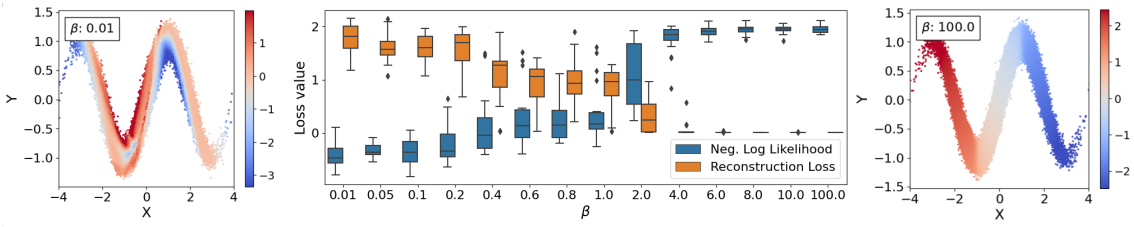


Figure 2.9: *Learning a noisy 2D sinusoid with a 1D latent space for different reconstruction weights β. Color codes in the left and right plots denote the value of the latent variable at each location. Box plots indicate variability across runs.*

Figure 2.9 demonstrates the behavior of the free-form injective flow objective as the reconstruction weight β varies. For low β values, the model tends to learn low-entropy directions, modeling only the noise perpendicular to the data manifold. This occurs because it results in a lower negative log-likelihood loss, while the higher reconstruction loss is insufficient to discourage this behavior. At even lower β values, the model becomes unstable.

As β increases, the model transitions to learning the data manifold effectively, successfully modeling the distribution on it. Between these two regimes, we observe a phase transition.

Tabular data

We evaluate our method on the four tabular distributions from Papamakarios et al. (2017), using the same data preprocessing and training splits. In Table 2.5, we compare our results to the published results for rectangular flows (RF) from Caterini et al. (2021). We adopt the "FID-like metric" used in their work, which computes the Wasserstein-2 distance between Gaussian distributions with equal mean and covariance as the test data and the model-generated data. This metric effectively measures the difference in means and covariance matrices between the generated and test datasets. Full experimental details and additional results can be found in Appendix E.2 of Sorrenson et al. (2024b).

Method	POWER	GAS	HEPMASS	MINIBOONE
RF (Caterini et al., 2021)	0.083 ± 0.015	$\mathbf{0.110 \pm 0.021}$	0.779 ± 0.191	1.001 ± 0.051
FIF (*ours*)	$\mathbf{0.041 \pm 0.007}$	0.281 ± 0.031	$\mathbf{0.541 \pm 0.034}$	$\mathbf{0.598 \pm 0.024}$
Training Time Speedup	**3.9** $\times$	**2.2** $\times$	**6.1** $\times$	**1.5** $\times$

Table 2.5: *Free-form injective flows (FIF) are significantly faster than rectangular flows (RF) with superior performance in FID-like metric (lower is better) on 3 out of 4 tabular datasets (Papamakarios et al., 2017). Both methods use $K = 1$. The results for RF are taken directly from (Caterini et al., 2021). Best results in bold.*

Our free-form injective flows (FIF) outperform RF on three out of four datasets (POWER, HEPMASS, and MINIBOONE), with the exception being the GAS dataset. Notably, FIF consistently demonstrates faster training times, with speedups ranging from 1.5× to 6.1× across all datasets. This combination of improved performance and significantly reduced training time highlights the efficiency and effectiveness of our approach in modeling complex tabular distributions.

Comparison to injective flows

Model	# parameters	$\mathcal{N}$ sampler		GMM sampler	
		FID ↓	IS ↑	FID ↓	IS ↑
DNF (Horvat and Pfister, 2021)	39.4M	55.6 ± 0.59	1.9	52.7 ± 0.33	2.0
Trumpet (Kothari et al., 2021)	19.1M	56.2 ± 1.39	1.8	47.7 ± 2.24	1.9
FIF (*ours*)	34.3M	$\mathbf{47.3 \pm 1.39}$	1.7	$\mathbf{37.4 \pm 1.35}$	2.0

Table 2.6: *Comparison of injective flows on CelebA under equal computational budget. Free-form injective flows (FIF) outperform previous work significantly in terms of FID (lower is better, best results in bold).*

We compare FIF to other injective flows on CelebA images (Liu et al., 2015) in Table 2.6. Our models achieve significantly better Fréchet Inception Distance (FID) (Heusel et al., 2017) and comparable Inception Score (IS) (Salimans et al., 2016). FID measures how well the distribution of generated samples matches the distribution of reference samples in the feature space of an image recognition model, while IS assesses the diversity of generated samples by calculating the entropy of outputs from a pretrained classification model.

All models are trained on the same hardware for equal wall clock time, ensuring a fair comparison. Further experimental details can be found in Appendix E.3 of Sorrenson et al. (2024b).

Comparison to generative autoencoders

The free-form injective flow (FIF) belongs to the broader category of generative autoencoders, which encompasses models such as variational autoencoders (VAEs) and their variants, as well as adversarial autoencoders (AAEs). To evaluate our model's performance relative to these approaches, we utilize the Pythae benchmark (Chadebec et al., 2022) for image generation tasks.

Model	Conv + $\mathcal{N}$ FID ↓	Res + $\mathcal{N}$ FID	Conv + GMM FID	Res + GMM FID
VAE	<u>54.8</u>	66.6	52.4	63.0
IWAE	55.7	67.6	52.7	64.1
VAE-lin NF	56.5	67.1	53.3	62.8
VAE-IAF	55.4	66.2	53.6	62.7
β-(TC) VAE	55.7	65.9	51.7	59.3
FactorVAE	**53.8**	66.4	52.4	63.3
InfoVAE - (RBF/IMQ)	55.5	66.4	52.7	62.3
AAE	59.9	<u>64.8</u>	53.9	58.7
MSSSIM-VAE	124.3	119.0	124.3	119.2
Vanilla AE	327.7	275.0	55.4	<u>57.4</u>
WAE - (RBF/IMQ)	64.6	67.1	51.7	57.7
VQVAE	306.9	140.3	<u>51.6</u>	57.9
RAE - (L2/GP)	86.1	168.7	52.5	58.3
FIF (*ours*)	56.9	**62.3**	**47.3**	**55.0**

Table 2.7: *Pythae benchmark results on CelebA, following Chadebec et al. (2022). We train their architectures (ConvNet and ResNet) with our new training objective, achieving SOTA FID on ResNet. We draw latent samples from standard normal "$\mathcal{N}$" or a GMM fit using training data "GMM". Models with multiple variants (indicated in brackets) have been merged to indicate only the best result across variants. We mark the best FID (lower is better) in each column in bold and underline the second best.*

The Pythae benchmark provides a standardized evaluation framework, assessing models across two different architectures on three datasets:

1. MNIST (LeCun et al., 2010): data dimension $D = 784$, latent dimension $d = 16$

2. CIFAR10 (Krizhevsky, 2009): $D = 3072$, $d = 256$

3. CelebA (Liu et al., 2015): $D = 12288$, $d = 64$

All models in the benchmark are allocated the same computational budget. It's important to note that due to these resource constraints, the benchmark's primary goal is not to produce state-of-the-art images, but rather to facilitate a fair comparison between methods under equal conditions.

Table 2.7 presents the results for the CelebA dataset, where our FIF model demonstrates superior performance, achieving the best overall results. The benchmark evaluates each architecture ("ConvNet" or "ResNet") using two sampling methods: drawing from a standard normal distribution ("$\mathcal{N}$") or from a Gaussian Mixture Model ("GMM") fitted post-training.

Our model also exhibits strong performance on the MNIST and CIFAR10 datasets. For a comprehensive view of the benchmark results across all datasets, along with training details and sample outputs from the models, we refer readers to Appendix E.4 in Sorrenson et al. (2024b).

Manifold	Dimension n	Embedding	Projection
Generic	$\text{rank}(\pi'(\pi(x)))$	$\{x \in \mathbb{R}^m : \pi(x) = x\}$	$x \mapsto \pi(x)$
Rotations $SO(d)$	$(d-1)d/2$	$\{Q \in \mathbb{R}^{d\times d} : QQ^T = \mathbb{I}, \det Q = 1\}$	$R \mapsto \arg\min_{Q\in SO(d)} \|Q - R\|_F$
Sphere $\mathbb{S}^n$	n	$\{x \in \mathbb{R}^{n+1} : \|x\| = 1\}$	$x \mapsto x/\|x\|$
Torus $\mathbb{T}^n = (\mathbb{S}^1)^n$	n	$\{X \in \mathbb{R}^{n\times 2} : \|X_i\| = 1 \text{ for } i = 1, \dots, n\}$	$X_i \mapsto X_i/\|X_i\| \text{ for } i = 1, \dots, n$
Hyperbolic $\mathbb{H}^n$	n	$\{x \in \mathbb{R}^n : \|x\| < 1\}$	$x \mapsto x \cdot \min\{1, (1-\epsilon)/\|x\|\}$

Table 2.8: *Manifolds, embeddings, and corresponding projections.*

2.4.3 Manifold free-form flows

Table 2.8 summarizes the manifolds used in our experiments, detailing their dimensions, embeddings, and projection functions. These diverse manifolds showcase the versatility of our manifold free-form flows (M-FFF) approach.

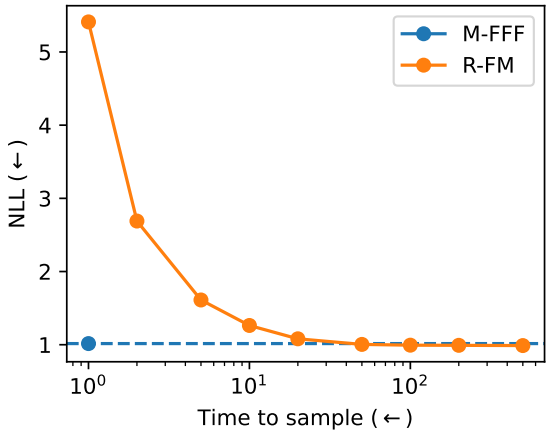


Figure 2.10: *At the same negative log-likelihood (NLL), M-FFF is 50 times faster than Riemannian Flow Matching (R-FM) (Chen and Lipman, 2024). At comparable inference speed, R-FM is significantly worse than M-FFF.*

Figure 2.10 illustrates the trade-off between speed and sample quality for M-FFF compared to other methods. While multistep methods such as Riemannian flow matching (R-FM) can achieve higher generative quality, especially for densities with sharp boundaries, they come at a significant computational cost—often 2 to 3 orders of magnitude more expensive. R-FM can trade off sampling time and quality by using an ODE solver with differing error tolerances, allowing for some flexibility in its performance, but performance degrades sharply at low sampling times. In contrast, M-FFF offers a consistently fast alternative without sacrificing too much quality.

The experiment shown in the figure uses the "general" protein backbone angle dataset (discussed in more detail in the section on torsion angles of molecules below). In this particular case, M-FFF demonstrates impressive efficiency, achieving comparable quality to R-FM while being approximately 50 times faster. This speed advantage allows for more extensive experimentation and potentially broader applications of M-FFF in scenarios where computational resources are limited or quick iterations are necessary.

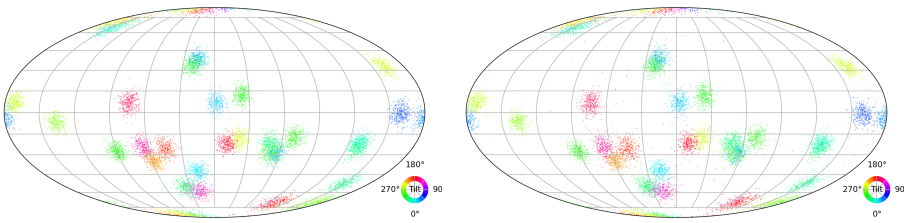


Figure 2.11: *Manifold free-form flows on a synthetic $SO(3)$ mixture distribution with $M = 64$ mixture components proposed by De Bortoli et al. (2022).* (Left) *10,000 samples each from the ground truth distribution and* (right) *our model. This visualization computes three Euler angles, which fully describe a rotation matrix, and then plots the first two angles on the projection of a sphere and the last by color (Murphy et al., 2021). We find that our model nicely samples from the distribution with few outliers between the modes.*

	$M = 16$	$M = 32$	$M = 64$	Fast inference?
Moser flow (Rozen et al., 2021)	-0.85 $\pm$ 0.03	-0.17 $\pm$ 0.03	0.49 $\pm$ 0.02	✗: 1000 steps
Exp-wrapped SGM (De Bortoli et al., 2022)	-0.87 $\pm$ 0.04	-0.16 $\pm$ 0.03	0.58 $\pm$ 0.04	✗: 500 steps
Riemannian SGM (De Bortoli et al., 2022)	**-0.89** $\pm$ **0.03**	-0.20 $\pm$ 0.03	0.49 $\pm$ 0.02	✗: 100 steps
$SO(3)$-NF (Liu et al., 2023)	-0.81 $\pm$ 0.01	-0.12 $\pm$ 0.004	0.61 $\pm$ 0.01	✓
M-FFF (*ours*)	**-0.87** $\pm$ **0.02**	**-0.21** $\pm$ **0.02**	**0.45** $\pm$ **0.02**	✓

Table 2.9: *Test negative log-likelihood (NLL, lower is better) on $SO(3)$ for multi-step and single-step methods. M-FFF consistently outperforms the specialized normalizing flow by Liu et al. (2023) on synthetic distributions of $SO(3)$ matrices and outperforms multi-step methods in the cases with more mixture components. Multi-step baseline values are due to De Bortoli et al. (2022). Best results in bold.*

Rotation matrices

$SO(3)$ is the group of 3D rotation matrices, defined as the set of matrices $Q \in \mathbb{R}^{3\times3}$ satisfying $QQ^T = \mathbb{I}$ and $\det(Q) = 1$. To project onto this manifold, we solve the constrained Procrustes problem by finding the orthogonal matrix closest to a given matrix in terms of Frobenius norm. This can be efficiently computed using the singular value decomposition (SVD) (Lawrence et al., 2019).

To evaluate our method on $SO(3)$, we use synthetic mixture distributions proposed by De Bortoli et al. (2022). These distributions consist of M mixture components, where we test with $M = 16, 32$, and 64. Our experiments demonstrate that our manifold free-form flows (M-FFF) can effectively model and sample from these complex distributions on $SO(3)$, as illustrated in Figure 2.11. As shown in Table 2.9, M-FFF achieves superior performance compared to both the specialized normalizing flow for $SO(3)$ developed by Liu et al. (2023) as well as the diffusion-based approaches with $M = 32$ and $M = 64$ components. Further details can be found in Appendix B.2 of Sorrenson et al. (2024a).

Earth data on the sphere

We train M-FFF to generate data on $\mathbb{S}^2$ using four earth science datasets compiled by Mathieu and Nickel (2020): volcanic eruptions (NGDC/WDS, 2022b), earthquakes (NGDC/WDS, 2022a), floods (Brakenridge, 2017), and wildfires (EOSDIS, 2020). As shown in Table 2.10, M-FFF significantly outperforms the specialized single-step model (Mixture

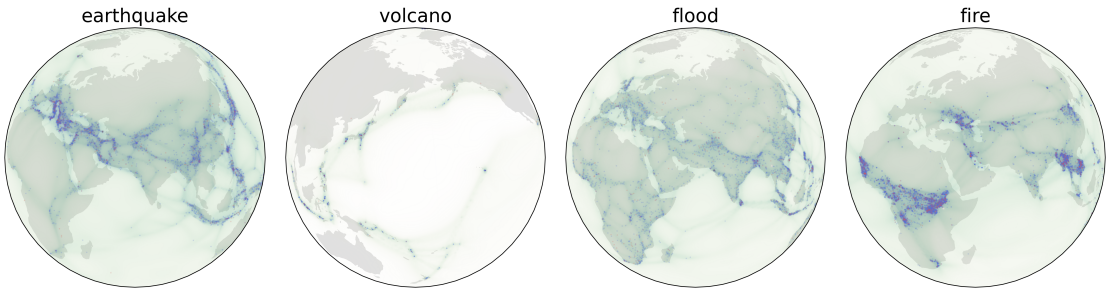


Figure 2.12: *Density estimates* (green) *of our model on the earth datasets. Blue points show the training dataset, red points the test dataset. Only one half of the sphere is shown.*

	Volcano	Earthquake	Flood	Fire	Fast inference?
Riemannian CNF (Mathieu and Nickel, 2020)	-6.05 ± 0.61	0.14 ± 0.23	1.11 ± 0.19	-0.80 ± 0.54	✗: ∼100 steps
Moser flow (Rozen et al., 2021)	-4.21 ± 0.17	-0.16 ± 0.06	0.57 ± 0.10	-1.28 ± 0.05	✗: ∼100 steps
Stereographic score-based (De Bortoli et al., 2022)	-3.80 ± 0.27	-0.19 ± 0.05	0.59 ± 0.07	-1.28 ± 0.12	✗: ∼100 steps
Riemannian score-based (De Bortoli et al., 2022)	-4.92 ± 0.25	-0.19 ± 0.07	0.45 ± 0.17	-1.33 ± 0.06	✗: ∼100 steps
Riemannian diffusion (Huang et al., 2022)	-6.61 ± 0.97	**-0.40 ± 0.05**	0.43 ± 0.07	-1.38 ± 0.05	✗: >100 steps
Riemannian flow matching (Chen and Lipman, 2024)	**-7.93 ±1.67**	-0.28 ± 0.08	**0.42 ± 0.05**	**-1.86 ± 0.11**	✗: 1000 steps
Mixture of Kent (Peel et al., 2001)	-0.80 ± 0.47	0.33 ± 0.05	0.73 ± 0.07	-1.18 ± 0.06	✓
M-FFF (*ours*)	-2.25 ± 0.02	**-0.23 ± 0.01**	**0.51 ± 0.01**	**-1.19 ± 0.03**	✓
Dataset size	827	6120	4875	12809	

Table 2.10: *M-FFF significantly outperforms the previous single-step density estimator (Peel et al., 2001) on the sphere on real-world earth datasets in terms of negative log-likelihood (lower is better). Baseline values are collected from De Bortoli et al. (2022); Huang et al. (2022); Chen and Lipman (2024). Best results in bold.*

of Kent). While M-FFF does not consistently outperform multi-step models in terms of log-likelihood, it offers a substantial speed advantage, being 2-3 orders of magnitude faster. Visual representations of the learned distributions are presented in Figure 2.12. Further details can be found in Appendix B.3 of Sorrenson et al. (2024a).

Torsion angles of molecules

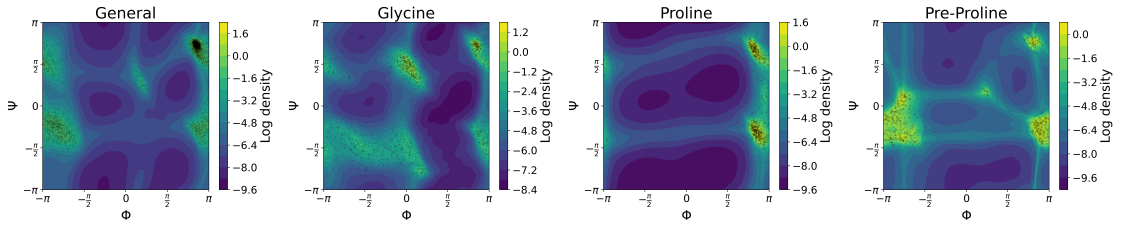


Figure 2.13: *Log density of M-FFF models in the* (Φ, Ψ)*-plane of protein backbone dihedral angles (known as the Ramachandran plot (Ramachandran et al., 1963)). The learned density matches the true density indicated by the test dataset* (black dots) *very well. Note also that the learned distribution obeys the periodic boundary conditions.*

We use tori to model the distribution of torsion angles in molecules. Since a torus in n dimensions is the product of n circles ($\mathbb{T}^n = (\mathbb{S}^1)^n$), a tuple of angles $(\phi_1, \ldots, \phi_n) \in [0, 2\pi]^n$ can be represented by mapping each angle to a position on a circle: $X_i = (\cos\phi_i, \sin\phi_i) \in \mathbb{S}^1$. These positions are then stacked into a matrix $X \in \mathbb{R}^{n\times 2}$.

We evaluate our method on two datasets from structural biology (Huang et al., 2022):

	General	Glycine	Proline	Pre-Pro	RNA	Fast inference?
Riemannian diffusion (Huang et al., 2022)	1.04 $\pm$ 0.012	1.97 $\pm$ 0.012	**0.12** $\pm$ 0.011	1.24 $\pm$ 0.004	-3.70 $\pm$ 0.592	✗: $\sim$1000 steps
Riemannian flow matching (Chen and Lipman, 2024)	**1.01** $\pm$ **0.025**	1.90 $\pm$ 0.055	0.15 $\pm$ 0.027	**1.18** $\pm$ **0.055**	**-5.20** $\pm$ **0.067**	✗: 1000 steps
Mixture of power spherical (Huang et al., 2022)	1.15 $\pm$ 0.002	2.08 $\pm$ 0.009	0.27 $\pm$ 0.008	1.34 $\pm$ 0.019	4.08 $\pm$ 0.368	✓
Circular Spline Coupling Flows (Rezende et al., 2020)	**1.03** $\pm$ **0.01**	1.91 $\pm$ 0.04	0.21 $\pm$ 0.08	1.24 $\pm$ 0.04	-4.01 $\pm$ 0.24	✓
M-FFF (ours)	**1.03** $\pm$ **0.02**	**1.89** $\pm$ **0.05**	**0.17** $\pm$ **0.08**	**1.23** $\pm$ **0.04**	**-4.27** $\pm$ **0.09**	✓

Table 2.11: *M-FFF consistently outperforms (in terms of negative log-likelihood, lower is better) normalizing flows specialized to tori (Rezende et al., 2020) on torus datasets, without requiring the development of a specialized architecture. In addition, our method comes close to the performance of the multi-step methods and even outperforms them on the Glycine dataset. Best results in bold.*

1. Ramachandran angles of protein backbones, located on $\mathbb{T}^2$ (Lovell et al., 2003). This dataset is split into four subcategories: General, Glycine, Proline, and Pre-Proline.
2. RNA structures characterized by 7 dihedral angles, located on $\mathbb{T}^7$ (Murray et al., 2003).

As shown in Table 2.11, M-FFF outperforms circular spline coupling flows, a specialized normalizing flow for tori (Rezende et al., 2020). Moreover, our results are comparable to those of multi-step methods, while being approximately three orders of magnitude faster.

For further details on the experimental setup and results, we refer the reader to Appendix B.4 of Sorrenson et al. (2024a).

Toy distributions in hyperbolic space

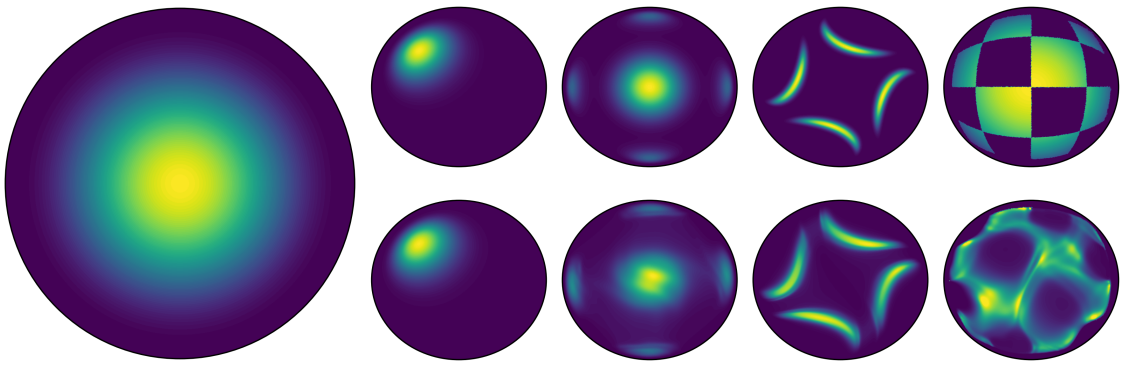


Figure 2.14: *Density estimation on the Poincaré ball model. As latent distribution, we use a wrapped normal distribution with standard deviation 0.5* (left). *As target distributions* (top row), *we define several toy distributions in the tangent space at the origin and push them forward to the manifold via the exponential map. We will reference each distribution from left to right as "one Gaussian," "five Gaussians," "swish," and "checkerboard." We train M-FFF on these target distributions and plot the densities of the models* (bottom row).

We use the Poincaré ball model to represent $\mathbb{H}^2$, the 2-dimensional hyperbolic space of constant negative curvature. This model embeds points in $\mathbb{H}^2$ as points inside the unit disc in $\mathbb{R}^2$, utilizing a projection from $\mathbb{R}^2$ into the unit disc (see Table 2.8 for details).

As there is no established benchmark for this task, we evaluate the performance of M-FFF by visualizing the learned densities (Figure 2.14). This approach allows us to confirm that the M-FFF framework successfully extends to non-isometrically embedded manifolds. Our results show that the learned densities closely match the target distributions for the

one Gaussian, five Gaussians, and swish datasets. However, for the checkerboard dataset, M-FFF struggles to fully reproduce the sharp edges and precise density patterns of the target distribution.

For a comprehensive description of the experimental setup and results, we refer readers to Appendix B.5 in Sorrenson et al. (2024a).

Manifold defined by mesh

	$k = 10$	$k = 50$	$k = 100$	Fast inference?
Riemannian Flow Matching (w/ diffusion) (Chen and Lipman, 2024)	1.16 $\pm$ 0.02	1.48 $\pm$ 0.01	1.53 $\pm$ 0.01	✗: 1000 steps
Riemannian Flow Matching (w/ biharmonic) (Chen and Lipman, 2024)	**1.06** $\pm$ **0.05**	1.55 $\pm$ 0.01	1.49 $\pm$ 0.01	✗: 1000 steps
M-FFF (*ours*)	**1.21** $\pm$ **0.01**	**1.34** $\pm$ **0.01**	**1.28** $\pm$ **0.01**	✓

Table 2.12: *Test negative log-likelihood (lower is better) on Stanford bunny data proposed by Chen and Lipman (2024), living on a manifold with nontrivial curvature (see Figure 2.5). M-FFF outperforms the multi-step model for datasets with more modes. Best results in bold.*

Our framework's flexibility extends to manifolds defined by meshes rather than explicit functional forms. We demonstrate this capability using the Stanford bunny dataset (Turk and Levoy, 1994), with training data generated from eigenvalues of the Laplace-Beltrami operator, following Chen and Lipman (2024). An example distribution is shown in Figure 2.5.

To ensure differentiability of the projection function, which is crucial for both the theoretical foundation and practical stability of our method, we train a neural network to project points from $\mathbb{R}^3$ onto the mesh.

As shown in Table 2.12, our method performs well on this task. We outperform Riemannian flow matching in two out of three cases, while being approximately three orders of magnitude faster in terms of inference speed. These results highlight the efficiency and effectiveness of our approach in modeling distributions on complex, mesh-defined manifolds. For more details, please refer to Appendix B.6 of Sorrenson et al. (2024a).

2.4.4 Summary of experimental results

Several key themes emerge from our experimental results:

- **Speed advantage:** The method consistently delivers inference speeds orders of magnitude faster than multi-step approaches.
- **Competitive performance:** Results generally match or exceed those of specialized architectures across different domains.
- **Flexibility:** The approach demonstrates remarkable adaptability across diverse manifolds and problem domains.
- **Trade-offs:** In some cases, the method trades modest performance decreases for substantial speed improvements.
- **Simplicity:** The implementation and tuning process is generally more straightforward than specialized architectures.

Chapter 3

Machine Learning in LHC Physics

During my PhD, I collaborated on several projects within Tilman Plehn's research group, focusing on applications of cutting-edge machine learning techniques to LHC data. My role was to suggest appropriate machine learning techniques for a given application, and to provide guidance that led to a successful implementation. While I did contribute some low-level implementation, particularly in the early stages of each project while prototyping models, most of the training and evaluation of the models was done by other students. Due to the nature of my contributions, this chapter will not include all training details necessary to reproduce the results. I will instead focus on higher-level concepts and guide those interested to the necessary sections of the corresponding research papers.

3.1 Particle physics at the Large Hadron Collider

Particle physics studies the fundamental particles of the universe and their interactions. The Large Hadron Collider (LHC) at CERN is a pivotal tool in this research, enabling scientists to probe the smallest scales of matter. This section will focus on the basic principles of particle physics explored at the LHC, with a particular emphasis on jet physics.

3.1.1 Fundamental particles and forces

The Standard Model of particle physics is the theoretical framework that describes the fundamental particles and their interactions. The primary particles include:

- **Quarks:** The building blocks of protons and neutrons, coming in six flavors: up, down, charm, strange, top, and bottom.
- **Leptons:** A family of particles that includes electrons, muons, tau particles, and their corresponding neutrinos.
- **Gauge bosons:** Force carriers that mediate the fundamental forces: photons (electromagnetic force), W and Z bosons (weak force), gluons (strong force), and the Higgs boson (mass generation).

Composite particles are also important, such as hadrons, made of quarks held together by the strong force, mediated by gluons. There are two main types of hadrons:

- **Baryons:** Made up of three quarks. Examples include protons and neutrons.
- **Mesons:** Made up of one quark and one antiquark. Examples include pions and kaons.

The LHC is called the Large Hadron Collider because it is designed to collide hadrons (specifically protons) and heavy ions (specifically lead ions) at extremely high energies, in a very large ring with a circumference of about 27 kilometers. The LHC accelerates these particles to nearly the speed of light and then collides them. These high-energy collisions can produce new particles and phenomena, providing insights into the fundamental laws of physics.

3.1.2 Particle collisions and jets

When particles are accelerated to high energies and collided at the LHC, they produce a variety of secondary particles. One of the key phenomena observed in these collisions is the formation of jets. Jets are collimated sprays of particles resulting from the hadronization of quarks and gluons. The study of jets is crucial for understanding the behavior of quarks and gluons under the strong force.

Jets are formed when high-energy quarks or gluons are produced in a collision and subsequently undergo hadronization, where quarks and gluons transform into hadrons due to the strong force. This process results in a narrow cone of particles moving in roughly the same direction, which is detected as a jet.

Jets are characterized by several properties that provide insights into the underlying physics of the collision. The jet energy represents the total energy of the particles within a jet. The jet mass, which is the invariant mass of the particles within a jet, can provide information about the original quark or gluon. Additionally, the jet shape, defined by the spatial distribution of particles within a jet, can indicate the type of parton (quark or gluon) that initiated the jet.

3.1.3 Applications of jet physics

Probing quantum chromodynamics (QCD)

Jets provide a direct way to study Quantum Chromodynamics (QCD), the theory of the strong interaction. By analyzing jet production rates, shapes, and energy distributions, physicists can test the predictions of QCD and improve our understanding of the strong force. Jets can also be used to study top quarks and the Higgs boson, as the jets produced in their decays provide crucial information about their properties and interactions.

Searches for new physics

Jets are also essential in searches for new physics beyond the Standard Model. Many theoretical models predict the production of new particles that decay into jets. By studying the properties of jets, physicists can search for deviations from the Standard Model that might indicate the presence of new particles or interactions.

3.2 Improved jet autoencoders

This section is based on work previously published as "Better latent spaces for better autoencoders" in SciPost Physics (Dillon et al., 2021). All figures are reproduced from that publication.

New physics searches at the LHC typically follow a signal-driven approach, where a specific signal is hypothesized and explicitly searched for within the data. However, if new physics does exist at the energy scales accessible by the LHC, it might be missed simply because we are searching for the wrong signals. A potential solution to this issue lies in unsupervised machine learning, which can uncover patterns from unlabeled data for further analysis. Of particular interest here is anomaly detection, also known as outlier detection.

A basic strategy for detecting outliers in jets involves using autoencoders, where the reconstruction error serves as the signal. By training a model to reconstruct data from a simulation of well-understood Standard Model processes, it is expected that this model will poorly reconstruct inputs that deviate from the Standard Model, flagging them as anomalies. While this method works well when trained on simpler QCD jets and tested on anomalous jets (e.g., top jets), it fails when tested in the reverse scenario (Heimel et al., 2019; Farina et al., 2020; Roy and Vijay, 2019; Blance et al., 2019). This suggests that the autoencoder's reconstruction loss primarily reflects the complexity of the jet structure rather than its novelty relative to the training data.

To address this issue, we explore autoencoders with more structured latent spaces, focusing specifically on the variational autoencoder (VAE) (Kingma and Welling, 2014). We consider three VAE models: one with a standard normal prior, another with a learnable two-component Gaussian mixture prior, and a third with a Dirichlet prior. Our findings indicate that the first two models struggle to effectively differentiate between QCD and top jets in an unsupervised manner. However, the VAE with a Dirichlet prior demonstrates a significant improvement. It successfully distinguishes these jet types even when the training set is dominated by either a small fraction of top jets or a small fraction of QCD jets, showing its robustness in scenarios with imbalanced datasets.

We use the QCD and top jet samples generated for the community top-tagging challenge outlined in Kasieczka et al. (2019), employing the same jet image representation as in Heimel et al. (2019). The average jet images for 100k QCD and top jets in this representation are displayed in Figure 3.1.

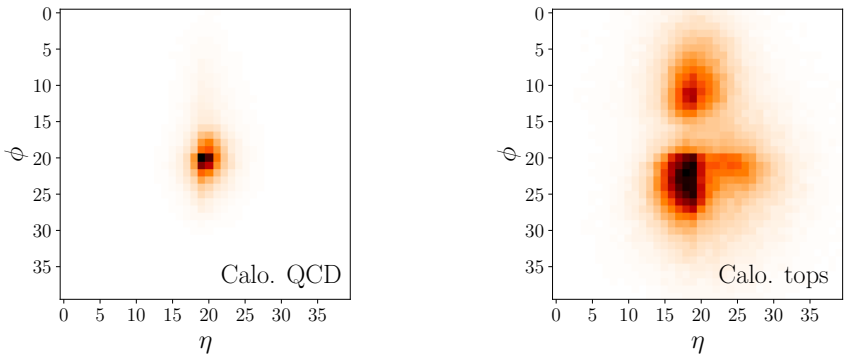
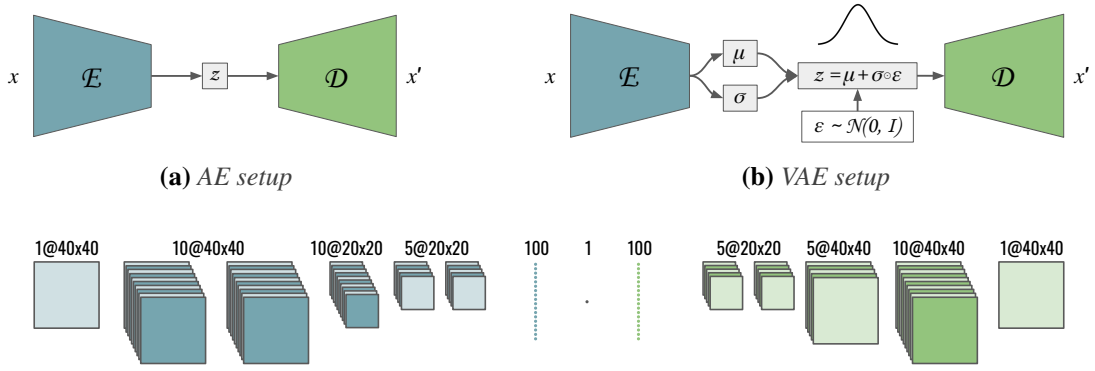


Figure 3.1: *Average of 100k QCD and top jet images after pre-processing.*

3.2.1 Autoencoder and variational autoencoder



(a) *AE setup* **(b)** *VAE setup*

(c) *Encoder and decoder*

Figure 3.2: *Architectures used for the AE and VAE networks in this study. All convolutions use a 5x5 kernel size and all layers use PReLU activations. Downsampling from 40x40 to 20x20 is achieved by average pooling, while upsampling is nearest neighbor interpolation.*

The autoencoder (AE) architecture consists of two networks: an encoder and a decoder, as visualized in Figure 3.2. We use the standard mean squared error as the loss function:

$$\mathcal{L} = \mathbb{E}_{p_{\text{data}}(x)} \left[\frac{1}{2} \|x - x'\|^2 \right] \tag{3.1}$$

where x' represents the reconstructed jet images from the input x.

Variational autoencoder

The variational autoencoder (VAE) has the same design as the AE, with the difference that the encoder has two outputs per dimension, corresponding to the mean and log-variance of a Gaussian distribution.

The VAE loss function follows the standard form (Kingma and Welling, 2014):

$$\mathcal{L} = \mathbb{E}_{p_{\text{data}}(x)} \left[\mathbb{E}_{q_\phi(z|x)} \left[-\log p_\theta(x|z) \right] + \beta_{\text{KL}} \mathcal{D}_{\text{KL}}(q_\phi(z|x) \parallel p(z)) \right] \tag{3.2}$$

where $q_\phi(z|x)$ is the learnable variational encoder and $p_\theta(x|z)$ is the decoder, with ϕ and θ representing their respective parameters. The β_{KL} term allows us to adjust the balance between the reconstruction loss (first term) and the latent loss (second term) during training. We use the mean squared error from Equation (3.1) as the reconstruction loss, consistent with a choice of Gaussian $p_\theta(x|z)$. For a standard normal prior $p(z)$, the KL divergence term has the following analytical form:

$$\mathcal{D}_{\text{KL}}(q_\phi(z|x) \parallel p(z)) = \frac{1}{2} \sum_{i=1}^{n} \left(\sigma_i^2 + \mu_i^2 - 1 - \log \sigma_i^2 \right) \tag{3.3}$$

where σ_i and μ_i are the outputs of the encoder for a given x. The network design is visualized in Figure 3.2.

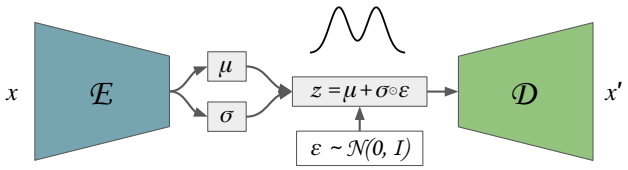


Figure 3.3: *Architecture used for the GMVAE. The encoder and decoder are the same as for the (V)AE shown in Figure 3.2c. In contrast to the standard VAE, the prior distribution is learnable and can be bimodal.*

3.2.2 Gaussian mixture VAE

In order to have a more expressive latent space where QCD and top jets have the possibility to be located in separate regions of the space, we introduce a Gaussian mixture prior with two mixture components. We approximate the KL divergence between the Gaussian mixture prior and the latent representation of the events using a Monte Carlo estimate. While several papers propose using a Gaussian mixture as the VAE prior (Dilokthanakul et al., 2016; Shao and Li, 2020; Yang et al., 2019; Guo et al., 2018), most implementations are more complex, often requiring additional inference networks to analytically compute the Kullback-Leibler (KL) divergence. The model most similar to ours (Shu et al., 2016) uses an approximation of the KL term (Hershey and Olsen, 2007) instead of a Monte Carlo estimate.

Likelihood function

We impose a Gaussian mixture prior on the n-dimensional latent space, where each Gaussian component in the mixture has a diagonal covariance. The conditional likelihood for this model is given by:

$$p(z|r) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_{r,i}} \exp\left(-\frac{(z_i - \mu_{r,i})^2}{2\sigma_{r,i}^2}\right) \tag{3.4}$$

where $\mu_{r,i}$ is the mean of the r-th mixture component (with $r = 1, \ldots, R$) in dimension i, and $\sigma_{r,i}^2$ is the corresponding variance. The prior is then calculated as $p(z) = \sum_r p(z|r)p(r)$, with $p(r)$ being the mixture weight of component r.

All means, variances, and mixture weights are learned parameters. The means are represented as a $R \times n$ matrix M, the log variances as a $R \times n$ matrix V, and the unnormalized log mixture weights as a vector a of length R. To ensure that the mixture weights are positive and sum to one, we apply the softmax operation:

$$p(r) = \text{softmax}(a_r) = \frac{e^{a_r}}{\sum_{r'} e^{a_{r'}}} \tag{3.5}$$

Using this notation, the conditional likelihood can be rewritten as:

$$p(z|r) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi e^{V_{ri}}}} \exp\left(-\frac{(z_i - M_{ri})^2}{2e^{V_{ri}}}\right) \tag{3.6}$$

In log form, this becomes:

$$\log\left(p(z|r)p(r)\right) = \sum_{i=1}^{n}\left(-\frac{(z_i - M_{ri})^2}{2e^{V_{ri}}} - \frac{1}{2}V_{ri}\right) - \frac{n}{2}\log(2\pi) + \log \text{softmax}(a_r) \quad (3.7)$$

from which we can apply the log-sum-exp operation to obtain the prior log-likelihood, $\log p(z) = \log \sum_r \exp \log(p(z|r)p(r))$. In our experiments, we always chose $R = 2$.

Loss function

As in a standard VAE, the GMVAE illustrated in Figure 3.3 minimizes the negative ELBO loss,

$$\mathcal{L} = \mathbb{E}_{p_{\text{data}}(x)}\left[-\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] + \beta_{\text{KL}}\mathcal{D}_{\text{KL}}\left(q_\phi(z|x) \parallel p(z)\right)\right] \quad (3.8)$$

with a learnable variational encoder $q_\phi(z|x)$ and decoder $p_\theta(x|z)$, where ϕ and θ represent the encoder and decoder parameters, respectively. The term β_{KL} allows control over the relative influence of the reconstruction and latent loss terms during gradient updates. Following the approach for the VAE and AE, we use mean squared error for the reconstruction loss and set $\beta_{\text{KL}} = 10^{-4}$.

The first term in Equation (3.8) represents the reconstruction loss, which is estimated by drawing a single sample from $q_\phi(z|x)$. The second term involves estimating the KL divergence, which becomes non-analytical for a Gaussian mixture prior. We estimate this divergence using Monte Carlo samples (Tomczak and Welling, 2018), starting by rewriting it as

$$\mathcal{D}_{\text{KL}}\left(q_\phi(z|x) \parallel p(z)\right) = \mathbb{E}_{q_\phi(z|x)}\left[\log q_\phi(z|x)\right] - \mathbb{E}_{q_\phi(z|x)}\left[\log p(z)\right] \quad (3.9)$$

The first term represents the negative entropy of a Gaussian, which can be computed analytically as

$$\mathbb{E}_{q_\phi(z|x)}\left[\log q_\phi(z|x)\right] = -\frac{n}{2}\left(1 + \log(2\pi)\right) - \frac{1}{2}\sum_{i=1}^{n}\log\sigma_i^2 \quad (3.10)$$

The second term in $\mathcal{D}_{\text{KL}}$ is estimated by drawing a single Monte Carlo sample from $q_\phi(z|x)$ and using the log-likelihood of the prior derived in Equation (3.7).

Mode separation loss

In testing the two-component mixture prior, we observed that the mixture components tend to collapse onto a single mode. To prevent this collapse, we introduce a repulsive force between the modes, calculated based on the Ashman distance (Ashman et al., 1994). The squared Ashman-D for two Gaussian distributions in one dimension is given by:

$$D^2 = \frac{2(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \quad (3.11)$$

A value of $D > 2$ indicates clear mode separation. This distance can be extended to two n-dimensional Gaussians with diagonal covariance matrices as:

$$D^2 = \sum_{i=1}^{n} \frac{2(\mu_{1i} - \mu_{2i})^2}{\sigma_{1i}^2 + \sigma_{2i}^2} \tag{3.12}$$

where, similarly, $D > 2$ indicates clear separation. To encourage bi-modality in the latent space, we add a loss term:

$$\mathcal{L}_A = -\lambda_A \tanh\left(\frac{D}{2}\right) \tag{3.13}$$

The tanh function eventually saturates, thereby stopping the modes from being pushed apart indefinitely.

3.2.3 Dirichlet VAE

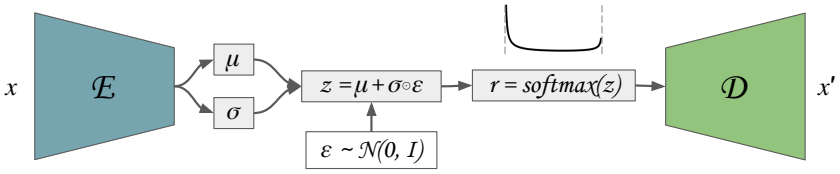


Figure 3.4: *Architecture for the Dirichlet VAE. The approximate Dirichlet prior is indicated by the softmax step and the bi-modal distribution shown above it.*

Finally, we explore an alternative approach to provide the VAE with a latent space structure that facilitates mode separation by using the Dirichlet distribution as the prior. The Dirichlet distribution is a family of continuous multivariate probability distributions with the following probability density function:

$$\mathcal{D}_\alpha(r) = \frac{\Gamma\left(\sum_i \alpha_i\right)}{\prod_i \Gamma(\alpha_i)} \prod_i r_i^{\alpha_i - 1}, \quad \text{with } i = 1, \ldots, R \tag{3.14}$$

This R-dimensional Dirichlet distribution is parameterized by R hyper-parameters $\alpha_i > 0$ and is defined on an R-dimensional simplex. The Dirichlet distribution is conjugate to the multinomial distribution and is commonly used in Bayesian statistics as a prior in multinomial mixture models. The expectation values for the sampled vector components are $\mathbb{E}[r_i] = \alpha_i / \sum_j \alpha_j$, thereby introducing a hierarchy among different mixture components when used as a prior. In our application, it imposes a compact latent space where the latent dimensions can be interpreted as mixture weights in a multinomial mixture model (Srivastava and Sutton, 2017; Joo et al., 2020).

Approximate Dirichlet distribution

For a Dirichlet structure in the latent space, the reparameterization trick requires careful consideration. We use a softmax Gaussian approximation to the Dirichlet distribution (Srivastava and Sutton, 2017), as Gaussian reparameterization is straightforward and stable:

$$r \sim \text{softmax}\,\mathcal{N}(z; \tilde{\mu}, \tilde{\sigma}) \approx \mathcal{D}_\alpha(r) \tag{3.15}$$

with

$$\tilde{\mu}_i = \log \alpha_i - \frac{1}{R} \sum_i \log \alpha_i \tag{3.16}$$

$$\tilde{\sigma}_i = \frac{1}{\alpha_i} \left(1 - \frac{2}{R}\right) + \frac{1}{R^2} \sum_i \frac{1}{\alpha_i} \tag{3.17}$$

With this approximation, the encoder network $q_\phi(r|x)$ functions similarly to how it does in the VAE and GMVAE, with the encoder outputs corresponding to the means and variances of the Gaussians in the softmax approximation.

Loss function

The loss function of the Dirichlet-VAE (DVAE) includes the standard reconstruction loss and latent loss. For the reconstruction loss, we use the cross-entropy between the inputs and the outputs (Srivastava and Sutton, 2017). The latent loss is defined as the KL divergence between the per-jet latent space representation and the Dirichlet prior, scaled by a factor β_{KL}. This divergence is straightforward to compute for the Gaussians in the softmax approximation of the Dirichlet distribution and the Gaussians defined by the encoder output (Srivastava and Sutton, 2017):

$$\mathcal{L} = \mathbb{E}_{p_{\text{data}}(x)} \left[-\mathbb{E}_{q_\phi(r|x)} \left[\log p_\theta(x|r) \right] + \beta_{\text{KL}} \mathcal{D}_{\text{KL}} \left(q_\phi(r|x) \parallel \mathcal{D}_\alpha(r) \right) \right] \tag{3.18}$$

$$\mathcal{D}_{\text{KL}} \left(q_\phi(r|x) \parallel \mathcal{D}_\alpha(r) \right) = \frac{1}{2} \sum_{i=1}^{R} \left(\frac{\tilde{\sigma}_i^2}{\sigma_i^2} + \frac{(\tilde{\mu}_i - \mu_i)^2}{\sigma_i^2} - R - \log \frac{\sigma_i^2}{\tilde{\sigma}_i^2} \right) \tag{3.19}$$

Unlike the GMVAE, the parameters of the prior in the DVAE are not learnable. We employ a straightforward DVAE architecture, as shown in Figure 3.4. The encoder consists of a fully connected network with a single hidden layer (no convolutional structure). The decoder is implemented as a simple linear function followed by a softmax activation.

3.2.4 Results

Latent-space tagging

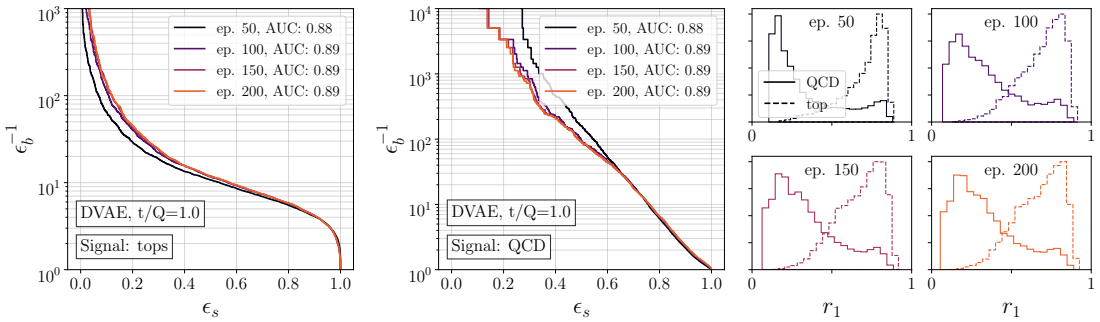


Figure 3.5: *Results for the DVAE. In the large panels we show the ROC curves for tagging top and QCD as signal. In the small panels we show the distributions of the top and QCD jets in the latent space developed over the training.*

We find that the VAE model with a standard normal latent distribution provides a more stable latent space than a simple autoencoder, though the performance improvement is

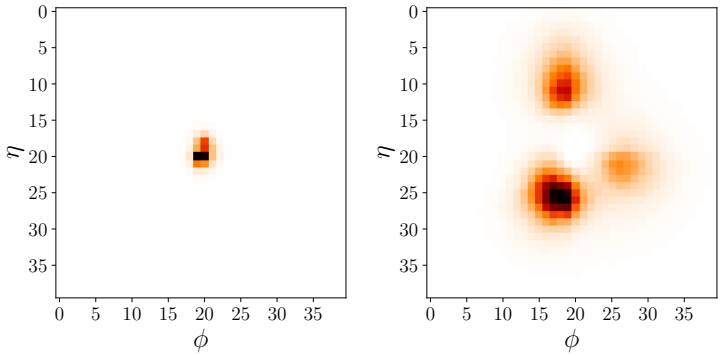


Figure 3.6: *Visualization of the decoder weights, interpreted as the mixture distributions for* $p_\theta(x|r_1=0)$ (left) *and* $p_\theta(x|r_1=1)$ (right) *learned by the DVAE decoder.*

marginal (see Section 2 and Figure 3 in Dillon et al. (2021)). The Gaussian mixture VAE (GMVAE) does achieve mode separation, but the clusters do not cleanly correspond to QCD and top jets—both types of jets are found within each cluster (see Section 3 and Figure 5 in Dillon et al. (2021)).

In contrast, the Dirichlet VAE (DVAE) performs significantly better on the same task. We use a 2D Dirichlet latent space with $\alpha = (1, 1)$, where due to the simplex nature of the Dirichlet distribution, only one latent space variable is independent. We set $\beta_{\text{KL}} = 0.1$ to ensure the prior strongly influences training. We compare QCD and top tagging results and show the latent space distributions in Figure 3.5.

The advantages of the DVAE are immediately evident. First, the performance improves from an AUC of 0.83 for the VAE to 0.89 for the DVAE. Second, the DVAE's latent space provides a better and more stable representation of the multi-class jet sample, quickly converging to a clear bi-modal pattern with both modes equally represented. The QCD mode peaks at $r_1 = 0$, indicating that the $p_\theta(x|r_1 = 0)$ mixture describes QCD jets, while the top mode peaks at $r_1 = 1$, indicating that the $p_\theta(x|r_1 = 1)$ mixture describes top jets. This is confirmed by visualizing the decoder weights in Figure 3.6. Comparing these to the truth-level jet images in Figure 3.1, we observe a clear correspondence between the learned mixture distributions and the actual images, showing that the DVAE has successfully separated the top-specific and QCD-specific features into distinct mixtures.

Anomaly detection

To move towards a more realistic scenario of unsupervised classification, we study cases with class imbalance, where one class is treated as anomalous. We slightly adjust our hyperparameters to reflect more hierarchical sub-class proportions (Dillon et al., 2020), setting $\alpha = (1.0, 0.25)$ when there is class imbalance in the data. The performance of the network is not overly sensitive to the exact values of these parameters. The advantage is that the DVAE tends to assign the dominant class of jets to $r_1 = 0$, thereby pushing anomalous jets towards $r_1 = 1$.

We evaluate four different signal-to-background ratios: 1.00, 0.25, 0.05, and 0.01, with the latter two representing anomalous cases. The top row of Figure 3.7 shows results for the scenario where the signal is top jets, while the bottom row shows results for QCD jets. We plot the ROC curves based on the latent space and reconstruction error, followed by the latent space distributions for the different signal-to-background ratios.

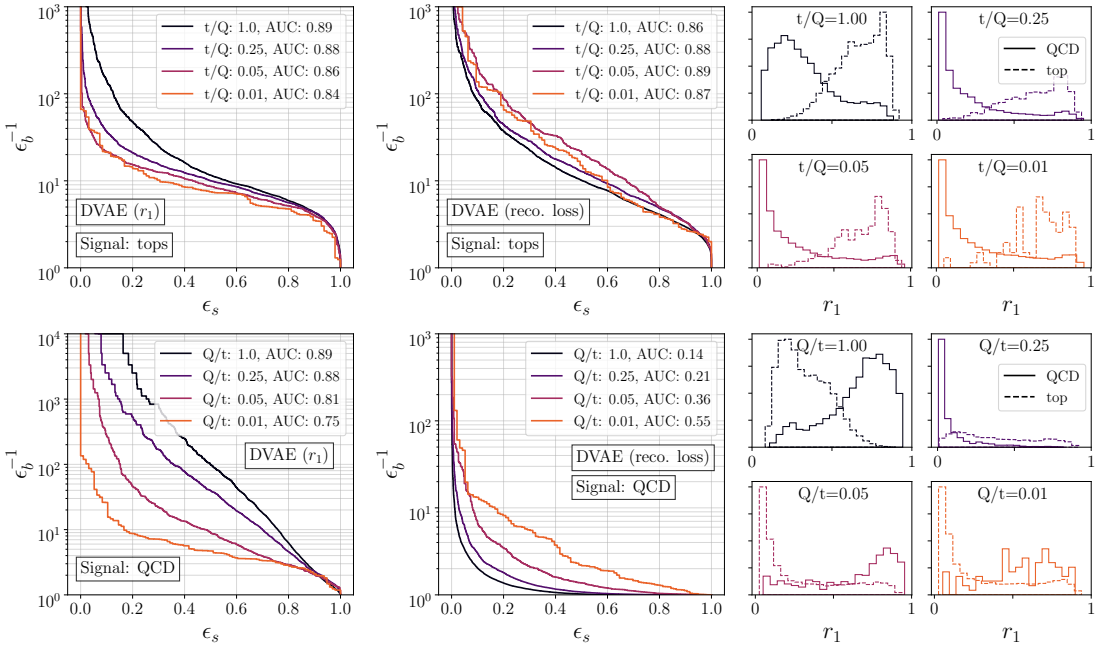


Figure 3.7: *DVAE results for various amounts of anomalous top* (upper) *and QCD* (lower) *jets in the sample, where the mixture weights* (left) *and the reconstruction loss* (middle) *are used for classification. In the small panels we show the distributions of the top and QCD jets in the latent space.*

When top jets are treated as the signal, classification based on mixture weights performs best at a signal-to-background ratio of $t/Q = 1.0$ and degrades as the ratio decreases. This is expected because the latent space needs sufficient information from a class to accurately construct a good latent representation. In contrast, the performance of the reconstruction loss improves as t/Q decreases, since the network typically reconstructs an underrepresented class poorly. This is why we observe reasonably good performance for anomalous top-tagging at $t/Q = 0.01$, consistent with findings in Heimel et al. (2019); Farina et al. (2020); Oleksiyuk (2020); Finke (2020).

The situation changes when QCD jets are treated as the signal. While latent space tagging remains stable with appropriate latent spaces, reconstruction error fails as a classifier. This reflects the motivation of our study and highlights the strength of our new approach.

The latent space distributions in Figure 3.7 confirm that when one class is anomalous, the Dirichlet prior effectively assigns the dominant class to the mixture component $r_1 = 0$. An exception is observed for $Q/t = 0.25$, where QCD jets are mistakenly assigned to $r_1 = 0$. This occurs because top jets can also have a strong central prong, mimicking typical QCD jet features. Since the unsupervised DVAE does not have access to truth labels and only clusters features, it assigns the dominant feature, which turns out to be QCD-like, even when $Q/t = 0.25$.

To better capture the differences between the two classes, particularly when treating QCD jets as anomalous, we expand the latent space to $R = 3$ with a hierarchical prior $\alpha_i = (1.0, 0.25, 0.1)$. This adjustment significantly enhances the performance of anomalous QCD tagging, increasing the AUC from 0.75 to 0.91. For a detailed analysis refer to Section 4 and Figures 11-13 in Dillon et al. (2021).

3.2.5 Summary

In this paper, we explored how different latent spaces in VAEs can enhance unsupervised jet classification and provide direct interpretability of what is learned by the neural networks. We began with a brief review of AEs and VAEs, discussing the limitations of both reconstruction error classification and latent space classification. Specifically, we highlighted the challenge of classifying anomalous jets with less structure than the dominant class in the sample, such as tagging anomalous QCD jets in a predominantly top-jet sample.

Our approach is motivated by the idea that autoencoders should leverage latent space structure to form modes representing distinct feature classes within the dataset, enabling background and signal separation. We first studied the Gaussian-Mixture-VAE, which indeed produced a multi-modal latent space, but the modes failed to capture all relevant feature classes. We then investigated the Dirichlet-VAE, which effectively identified hierarchical feature classes and organized jets in a way that separates signal from background. With a simple decoder architecture, we could interpret the decoder weights as mixture distributions corresponding to the latent space directions, allowing direct visualization and interpretation of what the network learns.

While the $R = 2$ Dirichlet latent space performed well for top-tagging, it did not symmetrically tag anomalous QCD jets in a predominantly top-jet sample. By extending the latent space to $R = 3$, the network successfully isolated QCD-like features, resulting in a significant improvement in classification. The larger latent space combined with a hierarchical prior enabled the extraction of features with varying prevalences in the dataset. With these structures, we demonstrated that the challenge of finding anomalous jets with less structure than the dominant class can be addressed by selecting more suitable latent space architectures. This opens the door to more effective jet anomaly detection techniques for collider analyses.

3.3 Representation learning for jets

This section is based on work previously published as "Symmetries, safety, and self-supervision" in SciPost Physics (Dillon et al., 2022). All figures are reproduced from that publication.

In collider physics, analyzing high-dimensional data is challenging because it is difficult to create representations that both respect fundamental physical symmetries and highlight important features. This work proposes JetCLR, a new approach using self-supervised contrastive learning to create data representations optimized for these goals. The technique involves a transformer-encoder network that can capture essential properties of jets while preserving key symmetries. When tested against other methods, JetCLR performs well in distinguishing different physics processes.

Symmetries play a central role in particle physics, guiding both theoretical models and experimental analyses. Traditional machine learning methods can produce powerful observables for specific tasks through supervised learning, but they often lack generality and cannot be easily applied to unsupervised tasks like anomaly detection. JetCLR takes a different approach by embedding symmetry principles directly into a high-dimensional representation. It uses contrastive learning, which clusters the data based on similarity without needing labels. The result is a flexible tool that can identify patterns in collider data while staying true to the underlying physics.

3.3.1 Existing jet representations

Existing jet representations in collider physics typically aim to respect the symmetries of the data while providing useful features for analysis. These include:

- **Jet images** are a common representation defined in terms of rapidity and azimuthal angle, often incorporating preprocessing to handle rotational symmetry (Cogan et al., 2015; de Oliveira et al., 2016; Kasieczka et al., 2017; Lin et al., 2018; Komiske et al., 2017; Macaluso and Shih, 2018).
- **Permutation-invariant graphs** represent jets as graphs, with nodes and edges capturing relationships between jet constituents (Henrion et al., 2017; Qasim et al., 2019; Chakraborty et al., 2019; Shlomi et al., 2020).
- **Tree-based structures** use tree-like architectures to represent jet evolution, capturing important jet features through hierarchical relationships (Louppe et al., 2019; Andreassen et al., 2019; Dillon et al., 2019, 2020).
- **Lund plane** represents jets in a plane that organizes splittings in terms of kinematic variables, providing a useful visualization for jet substructure (Dreyer et al., 2018; Carrazza and Dreyer, 2019).
- **Lorentz-inspired networks** are designed with Lorentz symmetry in mind, allowing for physics-inspired handling of relativistic jet features (Butter et al., 2018; Erdmann et al., 2019; Bogatskiy et al., 2020; Ju and Nachman, 2020; Shimmin, 2021).
- **Energy flow polynomials (EFPs)** provide a calculable basis for jet analysis, designed with infrared and collinear safety, making them theoretically grounded while retaining flexibility (Komiske et al., 2018).

3.3.2 Contrastive learning

In order to *learn* suitable representations for jets, we follow the SimCLR contrastive learning regime (Chen et al., 2020), originally proposed for representation learning on images.

The network outputs vectors z_i and z'_i, representing jets in $\mathbb{R}^{\dim(z)}$, which are normalized to lie on a unit hypersphere $\mathcal{R} = S^{\dim(z)-1}$, defining the representation as $f(x_i) = z_i/|z_i|$ and $f(x'_i) = z'_i/|z'_i|$. The similarity between jets is measured using cosine similarity, $s(z_i, z_j) = \cos\theta_{ij}$, where θ_{ij} is the angle between jets in $\mathcal{R}$. Since the representations lie on the unit hypersphere, the cosine similarity is simply the dot product. The so-called InfoNCE loss for a positive pair of jets is given by (van den Oord et al., 2018)

$$\mathcal{L}_i = -\log \frac{e^{s(z_i,z'_i)/\tau}}{\sum_{j\neq i}\left[e^{s(z_i,z_j)/\tau} + e^{s(z_i,z'_j)/\tau}\right]} \tag{3.20}$$

with the total loss $\mathcal{L} = \sum_i \mathcal{L}_i$ minimizing the distance between positive pairs while maximizing it between negative pairs. The hyperparameter τ controls the balance between positive and negative pairs.

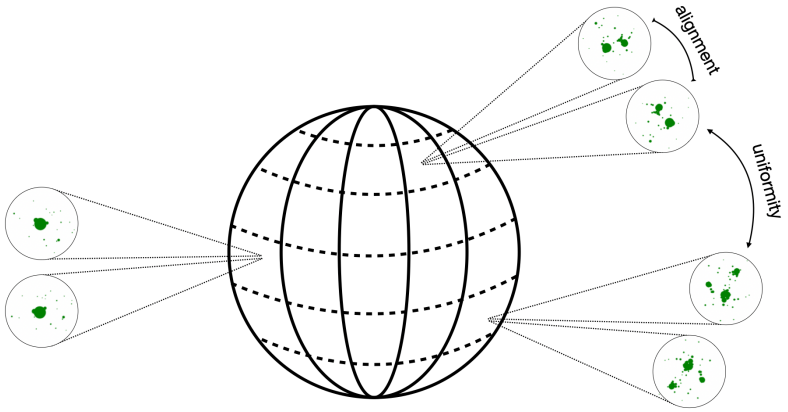


Figure 3.8: *Illustration of the uniformity and alignment concepts behind the contrastive learning.*

The contrastive loss can be interpreted as a balance between uniformity and alignment on the unit hypersphere $\mathcal{R}$, as shown in Figure 3.8. The numerator in Equation (3.20), related to positive pairs, is minimized when jets and their augmented versions are mapped to the same point, $s(z_i, z'_i) = 1$. Since they are constrained to the hypersphere, negative pairs cannot be pushed infinitely far apart, so the loss is minimized when jets are uniformly distributed across the hypersphere. Alignment and uniformity are measured by $\mathcal{L}_{\text{align}}$ and $\mathcal{L}_{\text{uniform}}$ respectively, with the former encouraging all jets and their augmented pairs to align, and the latter promoting uniform distribution:

$$\mathcal{L}_{\text{align}} = \frac{1}{N_{\text{batch}}} \sum_{i\in\text{batch}} s(z_i, z'_i) \tag{3.21}$$

$$\mathcal{L}_{\text{uniform}} = \frac{1}{N_{\text{batch}}} \sum_{i\in\text{batch}} \log \sum_{j\neq i} \left[e^{-s(z_i,z_j)} + e^{-s(z_i,z'_j)}\right] \tag{3.22}$$

While alignment alone would lead to a trivial solution where all points are identical, uniformity requires learning distinct features. The combination of both objectives creates

a representation space that is neither perfectly aligned nor perfectly uniform, balancing invariance and discriminative power.

3.3.3 Symmetries and augmentations

The mapping to the representation space $\mathcal{R}$ is designed to be approximately invariant to predefined symmetry transformations and data augmentations. Before applying these transformations in the contrastive learning process, jets are preprocessed such that their p_T-weighted centroid is positioned at the origin in the $\eta - \phi$ plane.

One key symmetry applied is rotational invariance around the jet axis. This is commonly handled in jet image representations through preprocessing, where the jet is rotated so its principal axis aligns vertically. Energy flow polynomials naturally account for rotational invariance by relying on angular distances between jet constituents. In our approach, jets are randomly rotated by angles sampled between 0 and 2π, a transformation that approximately preserves jet mass for narrow jets with $R \lesssim 1$.

In addition to rotations, translations in the $\eta - \phi$ plane are applied as a symmetry transformation. Each jet constituent is shifted by a random distance, with the shift constrained between -1 and 1 in both directions.

Beyond symmetry transformations, we incorporate theory-inspired augmentations based on the principles of quantum field theory. Soft gluon radiation, which factorizes from the hard physics in jet splittings, is modeled by smearing the positions of soft jet constituents. The η and ϕ coordinates are re-sampled from a Gaussian distribution centered on the original positions, with a variance of $\Lambda_{\text{soft}}/p_T$ where $\Lambda_{\text{soft}} = 100$ MeV.

Similarly, collinear splittings are accounted for through augmentations that reflect the detector's finite resolution, which prevents distinguishing closely spaced constituents. We simulate this effect by splitting some jet constituents while keeping the total p_T in an infinitesimal region constant. The resulting augmented jets retain the same η and ϕ coordinates but distribute the p_T randomly among the split constituents. Together, these soft and collinear augmentations introduce an approximate infrared and collinear (IRC) safety in the jet representation, allowing the contrastive learning process to optimize the mapping to $\mathcal{R}$ without explicitly enforcing fixed angular correlations or p_T scalings.

In summary, the transformations applied to the jets are:

- **Rotations**: Jets are rotated randomly by angles sampled from 0 to 2π in the $\eta - \phi$ plane, enforcing rotational symmetry and approximately preserving jet mass for narrow jets.
- **Translations**: Jet constituents are shifted by random distances within the range $[-1, 1]$ in both η and ϕ directions, introducing translational invariance.
- **Soft gluon smearing**: The positions of soft jet constituents are smeared by re-sampling their η and ϕ coordinates from a Gaussian distribution with a variance scaled by $\Lambda_{\text{soft}} = 100$ MeV and suppressed by p_T.
- **Collinear splitting**: Constituents are split while maintaining the total p_T in a small region, modeling the finite angular resolution of the detector and encoding collinear safety.

3.3.4 Network design

One of the key symmetries in jet representations is invariance to the ordering of constituents, i.e., permutation invariance. We directly encode this invariance into the network by using a transformer architecture (which is equivariant to permutations) followed by a summation over the constituent dimension.

We use a transformer architecture (Vaswani et al., 2017), with a decoder-only setup. The jet constituents x_i are first embedded into a higher-dimensional space through a linear projection, typically expanding to a dimension of 1000. Each transformer-encoder block then applies multi-headed self-attention, followed by a residual connection, layer normalization (Ba et al., 2016), and a feed-forward network that processes each constituent individually. This block is repeated N times, with a final layer normalization. The encoder outputs are summed over constituents to produce a fixed-size vector h, which is passed through a feed-forward head to generate the final output z. The architecture is shown in Figure 3.9. In practice, the linear classifier test shows h is a better representation than z, aligning with findings in the self-supervised literature (Chen et al., 2020).

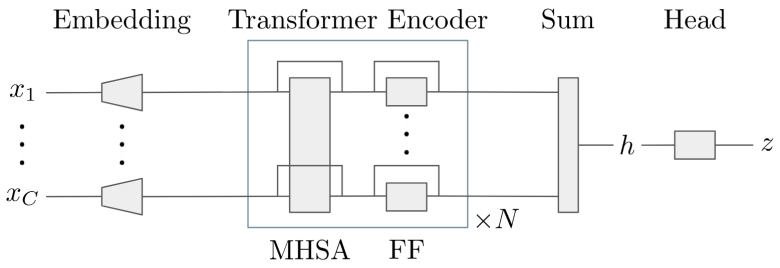


Figure 3.9: *Illustration of the transformer network architecture. MHSA stands for multi-headed self-attention, and FF denotes a feed-forward block.*

Jet constituents have a variable number per jet, so we apply zero-padding to jets with fewer constituents, allowing the batch to be converted into a single tensor for efficient computation. To prevent zero-padding from influencing the network output, masking is implemented in the transformer by setting attention weights for zero-valued constituents to negative infinity before softmax normalization. Additionally, zero constituents are excluded from the sum over constituents.

This masking approach can be generalized to be continuous in p_T. Instead of adding negative infinity, we add $\beta \log p_T$ to pre-softmax attention weights, and instead of setting some transformer outputs to zero, we multiply all outputs by the input p_T. This creates an IR-safe attention mechanism, ensuring that the transformer network is IR-safe by design.

Full architecture and training details can be found in our research paper (Dillon et al., 2022).

3.3.5 Results

To benchmark JetCLR, we investigate the contributions of symmetries and augmentations to performance, specifically focusing on top tagging, a well-studied area in the literature (Kasieczka et al., 2019). We employ the linear classifier test (LCT), a standard evaluation method in the machine learning community, to quantify performance. In this test, a

simple linear network is trained to classify between QCD and top jets, assessing the ability to separate the jets using a linear decision boundary in the representation space. We construct a ROC curve and utilize the inverse mistag rate at a given sensitivity (typically $\epsilon_b^{-1}(\epsilon_s = 0.5)$) and the area under the curve (AUC) as metrics. It is important to note that while ground truth labels are not used during the training of JetCLR via contrastive learning, they are employed during the evaluation of the LCT.

As dataset, we use QCD and top jet samples from the community top-tagging challenge (Kasieczka et al., 2019).

The representation is invariant to symmetries

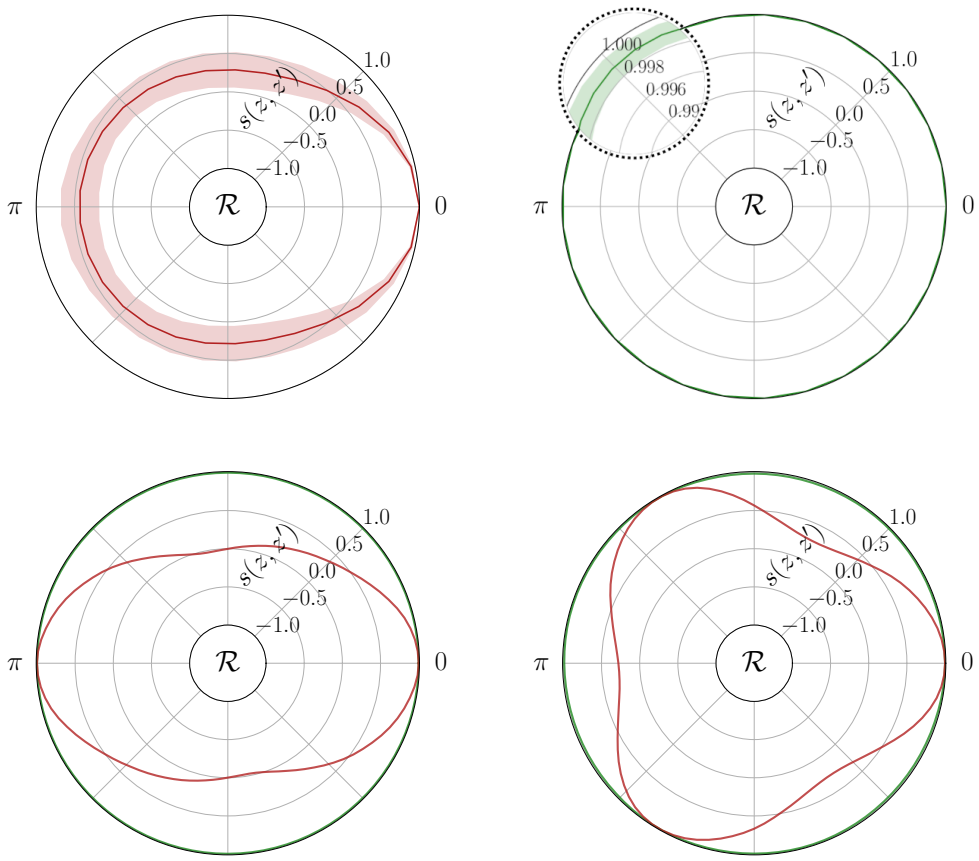


Figure 3.10: *Visualization of the rotational invariance in representation space, keeping in mind that* $s(z, z') = 1$ *indicates identical representations.* Top: *JetCLR representation trained without* (left) *and with* (right) *rotational transformations. Note the different scales of the radial axes showing the cosine similarity.* Bottom: *JetCLR representation for 2-prong* (left) *and 3-prong* (right) *toy jets, trained without* (red) *and with* (green) *rotational transformations.*

Of the two primary JetCLR tasks, invariance and discriminative power, we first confirm that the network correctly encodes symmetries. To demonstrate rotational symmetry, we assess how the representation remains invariant under actual jet rotations. Starting with a batch of 100 jets, we create rotated copies with angles evenly spaced between 0 and 2π. We then pass each jet and its rotated versions through the network, computing the cosine similarity between the original jet and its rotated copies. The top panels of Figure 3.10 show the mean and standard deviation of this similarity as a function of the rotation angle.

The scale of the radial axis $s(z, z')$ indicates that JetCLR trained with rotations produces representations more consistent with the original jets. In the left panel, similarity varies between 0.5 and 1.0, while in the right panel, the JetCLR representation is almost perfectly rotationally invariant.

Next, we test the rotational invariance of JetCLR representations using toy jets with $p_{T,j} = 600\,\text{GeV}$: one with two constituents and another with three equally spaced constituents, each sharing jet momentum equally. The lower panels of Figure 3.10 compare the rotational invariance of these toy jets. The red lines show the similarity functions for JetCLR representations of two-prong (left) and three-prong (right) jets without rotational training, where the maximum $s(z, z')$ values reflect geometric symmetry degeneracies. The green line shows the similarity functions for JetCLR representations trained with rotational transformations, confirming improved rotational invariance.

All augmentations in combination give the best results

Augmentation	$\epsilon_b^{-1}(\epsilon_s = 0.5)$	AUC
none	15	0.905
translations	19	0.916
rotations	21	0.930
soft+collinear	89	0.970
all combined	181	0.979

S/B	$\epsilon_b^{-1}(\epsilon_s = 0.5)$	AUC
1.00	181	0.980
0.50	160	0.979
0.25	150	0.978
0.10	161	0.978
0.05	146	0.978
0.01	158	0.978

Table 3.1: Left: *Classification results for JetCLR trained with different symmetries and augmentations and $S/B = 1$. The combined setup includes translation and rotation symmetries, combined with soft and collinear augmentations, and IR-safe masking.* Right: *Classification results for the combined symmetries and augmentations, trained with different signal-to-background ratio S/B.*

It is not straightforward to determine which symmetries and augmentations are most effective for learning JetCLR representations. The left panel of Table 3.1 summarizes the results after applying rotational, translational symmetry transformations, and soft+collinear augmentations. Among these, the soft+collinear augmentation performs best individually. While rotations and translations are less effective alone, combining all three leads to the best representations by a significant margin. These results were obtained using regular masking in the transformer. However, when combining all symmetries and augmentations, IR-safe masking slightly improves performance, so is included in the combined configuration.

Discriminative performance persists even when trained mostly on QCD jets

Although our initial results use a dataset with equal amounts of QCD and top jets, anomaly detection requires JetCLR to perform well even with fewer signal jets. The right panel of Table 3.1 shows how performance changes with a decreasing fraction of signal events in the training sample. Despite some expected noise, the results show that JetCLR's performance in the LCT is largely insensitive to the number of signal jets, indicating that JetCLR can capture essential structures based mainly on QCD jets and the applied symmetries and augmentations. This finding is promising for future anomaly searches using JetCLR representations.

JetCLR outperforms competitors

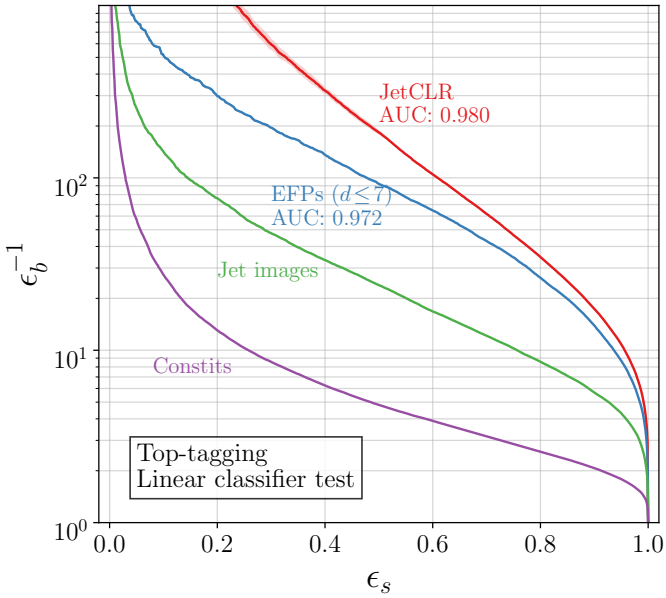


Figure 3.11: *Comparison of JetCLR with other classification metrics.*

After confirming that JetCLR encodes symmetries, we focus on the second task: evaluating discriminative power. To contextualize the results in Table 3.1, Figure 3.11 presents ROC curves comparing JetCLR to other representations. As expected, representations incorporating more physical symmetries perform better. Both the top-performing EFP and JetCLR representations use the same latent dimension, with the self-supervised JetCLR slightly outperforming the EFP representation in the linear classifier test (LCT) using binary cross-entropy loss. However, the performance difference is less clear when using other LCT variants. See the discussion in Dillon et al. (2022) for further details.

3.3.6 Summary

We used Contrastive Learning of Representations (CLR) to learn observables that respect symmetries and data augmentations while retaining discriminative power. Applied to jet physics, the JetCLR tool[1] uses a transformer-encoder network to encode rotation, translation, and permutation symmetries, and invariance under soft and collinear augmentations. Performance was assessed using a linear classifier test, showing that JetCLR outperforms simple jet images and competes with energy flow polynomials. The broader significance lies in demonstrating how symmetry principles and physics knowledge can be integrated into self-supervised ML tools. This opens promising directions for future applications, particularly in enhancing anomaly searches in LHC data.

[1]The JetCLR code is available at `https://github.com/bmdillon/JetCLR`

3.4 Generative models for jets I: Normalized autoencoder

This section is based on work previously published as "A normalized autoencoder for LHC triggers" in SciPost Physics Core (Dillon et al., 2023). All figures are reproduced from that publication.

The primary objective of the Large Hadron Collider (LHC) is to discover new physics beyond the Standard Model (BSM) and uncover new properties of the fundamental building blocks of matter. Traditionally, BSM searches have relied on predefined theoretical hypotheses, but in future LHC runs, we should complement these targeted searches with two new strategies: (i) analyses focusing on phase space regions connected to effective theory extensions of the Standard Model, and (ii) searches for anomalous effects that cannot be explained by the Standard Model. Both approaches aim to fully understand LHC data through fundamental physics, perhaps ultimately challenging the Standard Model.

A key concept in anomaly detection within LHC data is to make as few assumptions as possible about potential signals, in order to minimize bias and increase the chances of discovering genuinely new physics. Modern machine learning techniques, especially autoencoders (AEs), are promising in this regard, as they can, for example, identify anomalous jets in quantum chromodynamics (QCD) jet samples (Heimel et al., 2019; Farina et al., 2020).

However, AEs have their limitations. Since the latent space in AEs lacks a clear structure, the anomaly score is linked to the reconstruction quality. This can be problematic, especially when tagging anomalous top jets versus anomalous QCD jets, where one type of jet is considered signal and the other background. For instance, top jets stand out in a sample dominated by QCD jets, but a QCD jet does not appear particularly anomalous in a sample of top jets (Finke et al., 2021) because its simpler features can be easily interpolated by the neural network, despite not forming part of the training set.

To address these issues, we can use probabilistic models like variational autoencoders (VAEs) (Kingma and Welling, 2014), which can distinguish between high and low probability regions, or use other approaches that define anomalies based on low-probability regions in the background phase space distributions (Nachman and Shih, 2020; Batson et al., 2021; Dorigo et al., 2021; Caron et al., 2021; Fraser et al., 2021).

In this paper, we propose the normalized autoencoder (NAE) (Yoon et al., 2021) as a solution that combines the strengths of AEs and probabilistic models. The NAE uses an energy-based model (Xie et al., 2016; Du and Mordatch, 2019) defined by the reconstruction loss to identify outliers.

The NAE achieves reliable anomaly detection without increasing network complexity during inference; only the training process is more complex.

3.4.1 Energy-based networks

Energy-based models are a highly flexible class of models where the probability density is defined through a learnable energy function $E_\theta(x)$ as

$$p_\theta(x) \propto e^{-E_\theta(x)} \tag{3.23}$$

Note that the normalization constant is unknown and, in general, too computationally expensive to compute efficiently.

To train these models via maximum likelihood, we use the identity (see, e.g., Section 3 of Song and Kingma (2021))

$$\mathbb{E}_{x\sim p_{\text{data}}}\left[\nabla_\theta \mathcal{L}(x)\right] = \mathbb{E}_{x\sim p_{\text{data}}}\left[\nabla_\theta E_\theta(x)\right] - \mathbb{E}_{x\sim p_\theta}\left[\nabla_\theta E_\theta(x)\right] \tag{3.24}$$

where $\mathcal{L}(x) = -\log p_\theta(x)$. This means that in addition to sampling from $p_{\text{data}}(x)$ (which is straightforward), we must also sample from $p_\theta(x)$ (which is more challenging).

One practical approach to sample from $p_\theta(x)$ is to use Markov-Chain Monte Carlo (MCMC) methods. Specifically, we employ Langevin Markov Chains (Roberts and Tweedie, 1996), where the steps are defined by drifting a random walk towards high-probability regions according to

$$x_{t+1} = x_t + \lambda_x \nabla_x \log p_\theta(x) + \sigma_x \epsilon_t \qquad \text{with} \qquad \epsilon_t \sim \mathcal{N}(0,1) \tag{3.25}$$

Here, λ_x is the step size and σ_x is the noise standard deviation. When $2\lambda_x = \sigma_x^2$, this equation resembles Brownian motion and yields exact samples from $p_\theta(x)$ in the limits $t \to +\infty$ and $\sigma_x \to 0$.

In machine learning applications involving images, the high dimensionality of the data makes it challenging to fully explore the data space x with Markov chains of reasonable length. Consequently, it is common to use shorter chains and adjust λ and σ to emphasize the gradient term over the noise term. If $2\lambda \neq \sigma^2$, this approach corresponds to sampling from the distribution at a different temperature, defined as

$$T = \frac{\sigma^2}{2\lambda} \tag{3.26}$$

Here, λ and σ are defined as in Equation (3.25). By increasing λ or decreasing σ, we effectively sample from the distribution at a lower temperature, which allows for faster convergence towards the distribution's modes.

Despite the well-defined algorithm, training EBMs is notoriously difficult due to several factors: (i) the min-max optimization, which shares instability challenges similar to balancing a generator and discriminator in a GAN; (ii) potentially biased sampling from the MCMC due to a low effective temperature; and (iii) instabilities within the Langevin Markov Chains (LMCs). Altogether, stabilizing the training across its phases demands significant effort.

3.4.2 Normalized autoencoder

Autoencoders are not inherently probabilistic models. However, we can construct a probabilistic model by interpreting the reconstruction error as the energy, leading to what is known as the normalized autoencoder (NAE) (Yoon et al., 2021). In this framework, samples that are well-reconstructed have high likelihood, while those that are poorly reconstructed have low likelihood. By training the model using energy-based techniques, we ensure that inputs that would typically be well-reconstructed but do not appear in the training data (e.g., low-complexity inputs) are actually assigned high energy and, therefore, low probability.

The NAE is trained in two stages:

1. **Pretrain the base AE with MSE**: The model is first pretrained using the mean squared error (MSE) as the loss function. This stage is computationally cheap and helps the model assign low energy to the training data and high energy to high-complexity inputs not present in the training set.
2. **Switch to energy-based training with MSE as energy**: In this stage, the MSE is interpreted as the energy function, and the model undergoes energy-based training. This phase is more computationally expensive and refines the model, ensuring that low-complexity inputs only receive low energy if they are genuinely present in the data. During this stage, the model explores inputs close to, but not included in, the training dataset since it trains on samples from $p_\theta(x)$ as well as samples from $p_{\text{data}}(x)$. This contrasts with a standard AE, which would never encounter such samples during training and could exhibit unpredictable behavior outside of the training set.

Energy-based training

To train our model via energy, we take advantage of its autoencoder structure. First, we run Markov chains in the latent space, leveraging the fact that high-energy samples should be close to the decoder manifold—the set of points that can be generated by an input to the decoder. Since the latent space is lower-dimensional, this process is more efficient. Additionally, we ensure the latent space is compact by normalizing inputs to lie on the hypersphere, resulting in a sensible proposal distribution (uniform on the hypersphere) and a simplified search space. This strategy is known as on-manifold initialization (OMI) (Yoon et al., 2021).

The latent distribution is defined as

$$q_\theta(z) \propto e^{-E_\theta(f_D(z))} \tag{3.27}$$

where f_D is the decoding function.

After the latent space sampling, we proceed to run Markov chains in the full data space. In both the latent and data spaces, the chains are run at a high effective temperature, achieved by appropriately adjusting the parameters λ and σ. This strategy allows for efficient exploration of the relevant regions while minimizing the risk of spurious density peaks.

Further training details can be found in Section 2.2 of Dillon et al. (2023).

3.4.3 Results

Symmetric anomaly detection: QCD vs top jets

We evaluate our method on the well-established task of identifying top jets in a QCD background, as well as the symmetric task of identifying QCD jets in a top background. This scenario is particularly challenging for standard autoencoders, which struggle to identify QCD jets as anomalous when trained on a top jet background.

The dataset consists of jet images obtained from the top tagging challenge in the LHC Olympics (Kasieczka et al., 2021). These images were preprocessed following the methodology described in Heimel et al. (2019). Each jet image represents the energy deposits

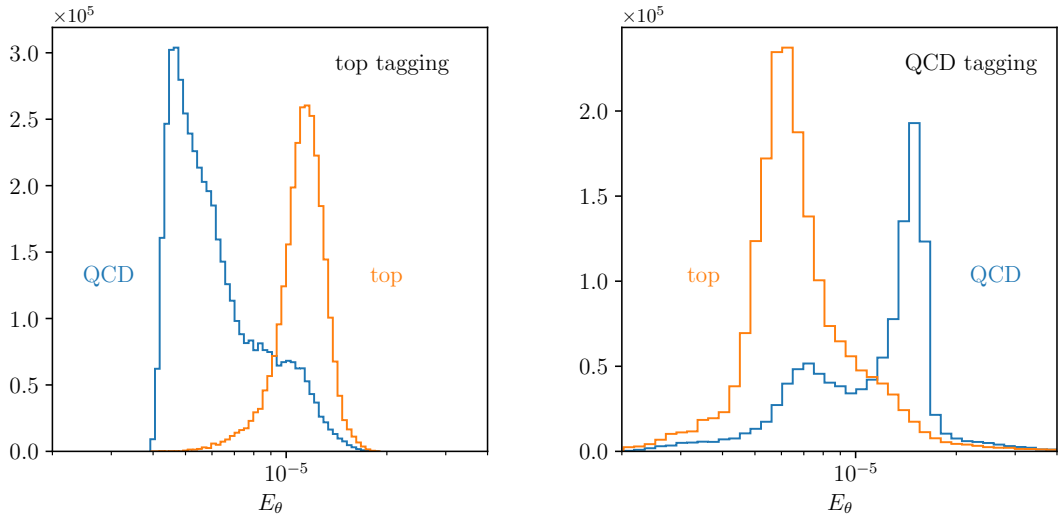
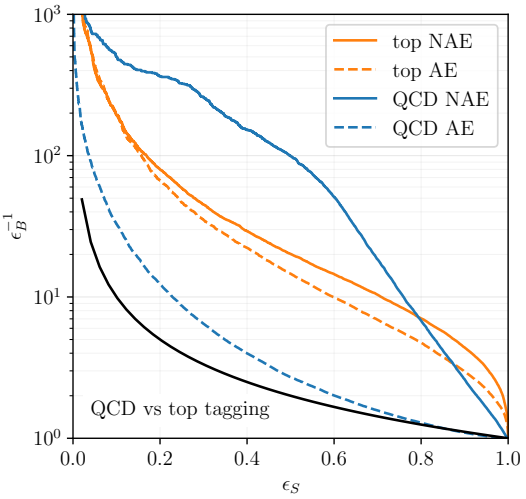


Figure 3.12: *Distribution of the energy (MSE) after training on QCD jets* (left) *and on top jets* (right). *We show the energy for QCD jets* (blue) *and top jets* (orange) *in both cases.*

in the calorimeter, providing a visual representation of the jet's structure and properties. A Gaussian filter is applied to the images to reduce their sparsity, making training more stable.



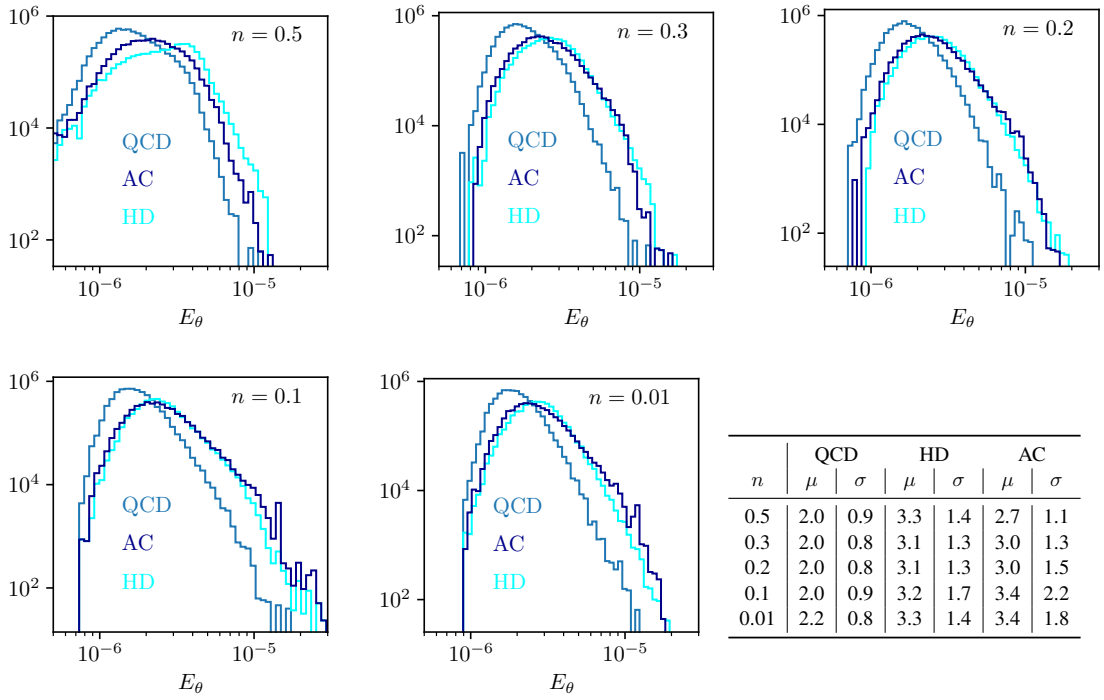
Signal	NAE	
	AUC	$\epsilon_B^{-1}(\epsilon_S = 0.2)$
top (AE)	0.875	68
top (NAE)	0.910	80
QCD (AE)	0.579	12
QCD (NAE)	0.890	350

Figure 3.13: *ROC curve for top* (orange) *and QCD* (blue) *tagging after AE pre-training* (dashed), *and after NAE training* (solid). *A random classifier corresponds to the solid black line. The table shows metrics extracted from the curves.*

Figure 3.12 displays the energy distributions for two models: one trained on QCD jets and another on top jets. The plots demonstrate that both models effectively differentiate between the two jet types, consistently assigning higher energy values to the background jets. In Figure 3.13, we present ROC curves that compare the performance of our NAE to a standard autoencoder (which is equivalent to the NAE after pre-training). The results show improved performance in both scenarios, with a particularly dramatic enhancement in the case where top jets form the background, where the AUC increases from 0.579 for the

standard autoencoder to 0.89 for the NAE. This substantial enhancement demonstrates the NAE's ability to overcome the limitations of standard autoencoders in anomaly detection tasks with complex, multi-modal data distributions.

Identifying dark jets in QCD backgrounds



	QCD		HD		AC	
n	μ	σ	μ	σ	μ	σ
0.5	2.0	0.9	3.3	1.4	2.7	1.1
0.3	2.0	0.8	3.1	1.3	3.0	1.3
0.2	2.0	0.8	3.1	1.3	3.0	1.5
0.1	2.0	0.9	3.2	1.7	3.4	2.2
0.01	2.2	0.8	3.3	1.4	3.4	1.8

Figure 3.14: *Distribution of the energy for QCD, Aachen, and Heidelberg datasets. Each panel corresponds to a different reweighting of the same datasets. The table shows the mean and the standard deviation for each distribution ($\times 10^{-6}$).*

Our second reference dataset comprises two signal samples inspired by dark matter models (Buss et al., 2022). These samples are based on hidden valley scenarios, featuring a light, strongly interacting dark sector (Strassler and Zurek, 2007; Morrissey et al., 2012; Knapen et al., 2021). In these models, particles produced in the dark sector can decay within it, forming a dark shower. This dark shower eventually transitions to Standard Model fragmentation, resulting in either a semi-visible jet (Cohen et al., 2015, 2017; Pierce et al., 2018; Beauchesne et al., 2018; Bernreuther et al., 2020, 2021) or a modified QCD jet (Heimel et al., 2019).

We refer to the semi-visible jets as the Aachen dataset and the modified QCD jets as the Heidelberg dataset. The Aachen dataset is characterized by its relatively sparse structure compared to the QCD background, while the Heidelberg dataset exhibits an additional decay structure. The QCD background for this analysis is generated using the same method as in the previous section, but with transverse momentum (p_T) ranging from 150 to 300 GeV and $|\eta| < 2$.

As with the top jets, we apply a Gaussian filter to the dark jet images, using $\sigma_G = 1$ to improve network training stability. Additionally, we implement a pixel-wise reweighting scheme to enhance the detection efficiency for both dark jet signals. This reweighting is

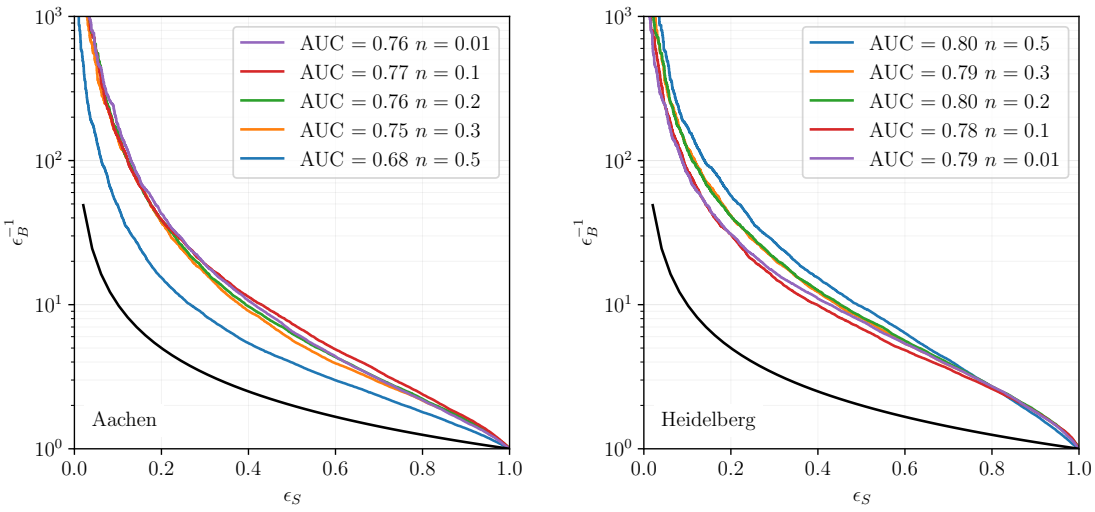


Figure 3.15: *ROC curve for dark jets tagging with different reweightings n, shown for the Aachen signal* (left) *and the Heidelberg signal* (right)*. A random classifier corresponds to the solid black line.*

applied uniformly across both dark jet types and is defined as:

$$p_T \to p_T^n \tag{3.28}$$

where n takes values of 0.01, 0.1, 0.2, 0.3, and 0.5. This approach aims to reduce the dependence of the autoencoder's performance on specific signal characteristics, allowing for more robust anomaly detection across different types of dark jets.

The effect of the reweighting is to alter the network's learning focus. At higher values of n, secondary clusters tend to be discarded, with the emphasis primarily on the main feature of QCD jets—the single prong structure. This makes it easier to identify the two-prong structure of Heidelberg jets. Conversely, when n is low, secondary clusters are enhanced and not easily discarded. The main prong is not learned as well, and the sparse structure of the Aachen jets is emphasized. For a more detailed discussion of this trade-off, refer to Section 4 of Dillon et al. (2023).

Figure 3.14 presents the energy distributions of the QCD, Aachen, and Heidelberg datasets as histograms for different reweighting values of n. We observe that both dark jet datasets consistently exhibit higher energy than the QCD jets. The performance remains relatively stable across different n values, with a significant difference only appearing at $n = 0.5$. In this case, the Heidelberg dataset becomes more distinguishable from the background. However, as shown in Figure 3.15, the performance for the Aachen dataset decreases at this n value. The optimal overall performance is achieved with $n = 0.2$, balancing the trade-offs between different jet structures.

3.4.4 Summary

In summary, our experiments with the Normalized Autoencoder (NAE) demonstrate its effectiveness in symmetric anomaly detection for jet physics. In the QCD vs. top jet

experiment, the NAE showed remarkable performance in both directions of tagging. When trained on QCD jets to detect anomalous top jets, it achieved an AUC of 0.91, surpassing previous autoencoder benchmarks. More importantly, when trained on top jets to detect anomalous QCD jets—a task that traditional autoencoders struggle with—the NAE achieved a competitive AUC of 0.89. This symmetric performance highlights the NAE's ability to overcome the limitations of standard autoencoders in handling datasets of varying complexity.

For the dark jet experiments, the NAE demonstrated its versatility in detecting two distinct types of dark jets—the Aachen (semi-visible) and Heidelberg (modified QCD) datasets—against a QCD background. By implementing a pixel-wise reweighting scheme, we were able to fine-tune the NAE's sensitivity to different jet structures. The reweighting parameter n allowed us to balance the detection efficiency between the sparse Aachen jets and the more structured Heidelberg jets. Optimal performance was achieved at $n = 0.2$, where the NAE showed good discrimination for both dark jet types. This adaptability underscores the NAE's potential as a robust tool for anomaly detection in various jet physics scenarios, capable of handling different signal characteristics within a single framework.

3.5 Generative models for jets II: Diffusion models and JetGPT

This section is based on work previously published as "Jet Diffusion versus JetGPT — Modern Networks for the LHC" on arXiv (Butter et al., 2023). All figures are reproduced from that publication.

3.5.1 Generative models at the LHC

Simulation-based analysis plays a crucial role in research conducted at the Large Hadron Collider (LHC). The simulation chain encompasses a range of processes, from hard scattering to detector simulations. To be effective, this chain must maintain both precision and computational efficiency.

In recent years, generative neural networks have emerged as the leading machine learning tools for LHC simulations (Butter and Plehn, 2020; Butter et al., 2022). While variational autoencoders (VAEs) and generative adversarial networks (GANs) have shown promise, they often lack the necessary precision for high-energy physics applications. Normalizing flows, particularly those utilizing invertible coupling blocks, have demonstrated success in lower-dimensional problems such as hard scattering simulations.

This study aims to expand upon the current state of the art by comparing normalizing flows to three advanced generative models: two based on diffusion processes (Sohl-Dickstein et al., 2015; Ho et al., 2020; Albergo and Vanden-Eijnden, 2023; Lipman et al., 2023; Liu et al., 2022) and one leveraging an autoregressive transformer architecture (Radford et al., 2018). By evaluating these cutting-edge approaches, we seek to identify potential improvements in both the accuracy and efficiency of LHC simulation techniques.

3.5.2 Denoising diffusion probabilistic model

The denoising diffusion probabilistic model (DDPM) is a generative model that learns to gradually denoise data from a simple noise distribution to the target data distribution. The process consists of two main steps: a forward diffusion process and a reverse denoising process.

Forward diffusion process

In the forward diffusion process, we start with a data point x_0 and gradually add Gaussian noise over T timesteps:

$$\begin{aligned} x_1 &\sim \mathcal{N}(\sqrt{1-\beta_1}x_0, \beta_1 I) \\ x_2 &\sim \mathcal{N}(\sqrt{1-\beta_2}x_1, \beta_2 I) \\ &\vdots \\ x_T &\sim \mathcal{N}(\sqrt{1-\beta_T}x_{T-1}, \beta_T I) \end{aligned} \tag{3.29}$$

where β_t is a noise schedule that typically increases with t. This process can be expressed in closed form for any arbitrary timestep t:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \tag{3.30}$$

where $\bar{\alpha}_t = \prod_{i=1}^{t}(1-\beta_i)$ and $\epsilon \sim \mathcal{N}(0, I)$.

Reverse denoising process

The reverse process aims to learn how to denoise the data, starting from pure noise x_T and gradually recovering the original data x_0. This is modeled as a series of Gaussian transitions:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{3.31}$$

where θ are the parameters of a neural network that predicts the mean and variance of this Gaussian distribution. In practice, it is sufficient to set $\Sigma_\theta(x_t, t) = \beta_t \mathbb{I}$.

Training the model

The model is trained to minimize the negative log-likelihood of the reverse process. Up to some constants which don't depend on the model parameters, and discarding prefactors, this is equivalent to minimizing the following loss:

$$\mathcal{L} = \mathbb{E}_{x_0,\epsilon,t}[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|^2] \tag{3.32}$$

where ϵ_θ is a noise prediction network. For further details see Ho et al. (2020).

Retaining prefactors yields a more precise loss function:

$$\mathcal{L} = \mathbb{E}_{x_0,\epsilon,t}\left[\frac{1}{2\sigma_t^2}\frac{\beta_t^2}{(1-\beta_t)\bar{\beta}_t}\|\epsilon - \epsilon_\theta(\sqrt{1-\bar{\beta}_t}x_0 + \sqrt{\bar{\beta}_t}\epsilon, t)\|^2\right] \tag{3.33}$$

where $\sigma_t^2 = \beta_t$ and $\bar{\beta}_t = 1 - \bar{\alpha}_t = 1 - \prod_{i=1}^{t}(1-\beta_i)$.

We use this variant because it:

1. Is an exact likelihood loss, adaptable to Bayesian settings
2. Maintains or improves model performance
3. Provides more accurate loss weighting across timesteps

Inference

During inference, we can sample from the model by starting with pure noise $x_T \sim \mathcal{N}(0, \mathbb{I})$ and iteratively applying the learned denoising process until we obtain x_0.

Bayesian version of DDPM

To create a Bayesian version of the DDPM, we replace the deterministic noise prediction network with a Bayesian neural network (BNN). This BNN is trained by minimizing the Kullback-Leibler (KL) divergence between a variational approximation $q(\theta)$ and the posterior distribution $p(\theta|\text{data})$, where θ represents the network parameters. The variational approximation $q(\theta)$ serves as a tractable proxy for the true posterior, while $p(\theta)$ represents a prior distribution over the network weights.

This Bayesian approach offers several advantages:

1. **Uncertainty quantification:** It provides a measure of uncertainty in the model's predictions, which is crucial for scientific applications.

2. **Regularization:** The prior $p(\theta)$ acts as a form of regularization, potentially improving generalization.

3. **Model averaging:** At inference time, we can sample multiple sets of weights from $q(\theta)$, effectively performing model averaging.

The loss function for this Bayesian DDPM becomes:

$$\mathcal{L} = \mathbb{E}[\log p(x|\theta)] - \text{KL}(q(\theta) \parallel p(\theta)) \tag{3.34}$$

where the first term is the expected log-likelihood of the data given the model parameters, and the second term is the KL divergence between the variational posterior and the prior. This formulation allows us to balance between fitting the data and maintaining a simple model, as determined by the prior.

Full training and architecture details can be found in Section 2.1 of Butter et al. (2023).

3.5.3 Conditional flow matching

Conditional Flow Matching (CFM) is another approach that transforms data to noise in a forward process and then generates from noise in a backward process. Unlike DDPM, the CFM model is continuous, and the backward pass is solved as an ordinary differential equation (ODE), similarly to a continuous normalizing flow (Chen et al., 2018). In this method, we learn a velocity field that can be understood as a weighted integral over many conditional velocity fields, hence the name "conditional flow matching."

Loss function derivation

We begin with an important identity:

$$\mathbb{E}_{p(y|x)}[\|f(x) - y\|^2] = \|f(x) - \mathbb{E}_{p(y|x)}[y]\|^2 + \text{const.} \tag{3.35}$$

where the constant is independent of f.

The core of the CFM approach is based on the continuity equation:

$$\frac{\partial p(x,t)}{\partial t} = -\nabla_x \cdot [p(x,t)v(x,t)] \tag{3.36}$$

where $p(x,t)$ is the probability density at point x and time t, and $v(x,t)$ is the velocity field.

We define a conditional probability $p(x,t|x_0)$ which we generate from by:

$$x = (1-t)x_0 + t\epsilon \tag{3.37}$$

The associated conditional velocity is:

$$v(x,t|x_0) = \epsilon - x_0 \tag{3.38}$$

Then we use that $p(x,t|x_0)$ and $v(x,t|x_0)$ satisfy the continuity equation to derive the total velocity field which satisfies the continuity equation for the total probability $p(x,t)$:

$$\frac{\partial p(x,t)}{\partial t} = \int \frac{\partial p(x,t|x_0)}{\partial t} p(x_0)\,\mathrm{d}x_0 \tag{3.39}$$

$$= -\int \nabla_x \cdot [p(x,t|x_0)v(x,t|x_0)]p(x_0)\,\mathrm{d}x_0 \tag{3.40}$$

$$= -\nabla_x \cdot \left[p(x,t)\int v(x,t|x_0)p(x_0|x,t)\,\mathrm{d}x_0\right] \tag{3.41}$$

$$= -\nabla_x \cdot [p(x,t)v(x,t)] \tag{3.42}$$

where

$$v(x,t) = \int v(x,t|x_0)p(x_0|x,t)\,\mathrm{d}x_0 = \mathbb{E}_{p(x_0|x,t)}[v(x,t|x_0)] \tag{3.43}$$

The loss function for training the CFM model can be derived as follows:

$$\mathbb{E}_{p(x,t)}[\|v_\theta(x,t) - v(x,t)\|^2] = \mathbb{E}_{p(x,t)}[\|v_\theta(x,t) - \mathbb{E}_{p(x_0|x,t)}[v(x,t|x_0)]\|^2] \tag{3.44}$$

$$= \mathbb{E}_{p(x,t)}[\mathbb{E}_{p(x_0|x,t)}[\|v_\theta(x,t) - v(x,t|x_0)\|^2] + \text{const.}] \tag{3.45}$$

$$= \mathbb{E}_{p(x_0,x,t)}[\|v_\theta(x,t) - v(x,t|x_0)\|^2] + \text{const.} \tag{3.46}$$

This formulation allows us to train the model by minimizing the expected squared difference between the predicted velocity field $v_\theta(x,t)$ and the true conditional velocity field $v(x,t|x_0)$.

This gives the loss function:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{p(x_0,\epsilon,t)}[\|v_\theta((1-t)x_0 + t\epsilon, t) - (\epsilon - x_0)\|^2] \tag{3.47}$$

where v_θ is our learned velocity field, x_0 is the initial data point, ϵ is sampled from the noise distribution, and t is uniformly sampled from $[0,1]$.

Advantages of CFM

The CFM approach offers several advantages:

1. It provides a continuous-time formulation, allowing for flexible sampling schemes.
2. The backward process is deterministic, solved via an ODE, which can be more efficient than the stochastic process in DDPM.
3. The loss function is simpler and more intuitive compared to DDPM, potentially leading to easier optimization.

Full training and architecture details for our implementation of CFM can be found in Section 2.2 of Butter et al. (2023). That section also includes information on how to create a Bayesian version of CFM.

3.5.4 Autoregressive transformer

Autoregressive models learn to generate data one dimension at a time. They factorize the joint probability distribution as a product of conditional probabilities:

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1}) \tag{3.48}$$

This approach breaks down the problem of learning an n-dimensional joint distribution into learning n one-dimensional conditional distributions. Each of these conditional distributions is modeled as a function of the previous dimensions.

Parameter prediction

We can implement this by predicting the parameters of a simple one-dimensional model for each dimension, conditioned on the previous dimensions. For example, we might predict:

- Probabilities of discrete bins
- Means and variances of a Gaussian mixture model (GMM)

Formally, we can express this as:

$$\omega^i = f_\theta^i(x_1, \ldots, x_{i-1}) \tag{3.49}$$

where ω^i represents the parameters for the i-th dimension, and f_θ^i is a neural network with parameters θ. The conditional probability for the i-th dimension is then given by:

$$p(x_i | x_1, \ldots, x_{i-1}) = p(x_i | \omega^i) \tag{3.50}$$

Loss function

The loss function for training such a model is typically the negative log-likelihood:

$$\mathcal{L} = -\mathbb{E}_{x \sim p_{\text{data}}}[\log p(x)] = -\mathbb{E}_{x \sim p_{\text{data}}} \left[\sum_{i=1}^{n} \log p(x_i | x_1, \ldots, x_{i-1}) \right] \tag{3.51}$$

This loss function encourages the model to assign high probability to the observed data points, thereby learning the underlying data distribution.

Sampling

Sampling from the model proceeds sequentially, dimension by dimension. For each dimension i, we:

1. Compute the parameters $\omega^i = f_\theta^i(x_1, \ldots, x_{i-1})$
2. Sample $x_i \sim p(x_i | \omega^i)$

This process continues until all dimensions have been sampled, resulting in a complete sample $x = (x_1, \ldots, x_n)$.

Transformer implementation

To implement the functions f_θ^i, we use a transformer encoder architecture with causal masking. This approach is similar to the GPT (Generative Pre-trained Transformer) models used in natural language processing (Radford et al., 2018, 2019). The causal masking ensures that each prediction only depends on the previous dimensions, maintaining the autoregressive property.

The transformer architecture allows for efficient parallel computation during training, as all f_θ^i can be computed simultaneously. During sampling, however, we must proceed sequentially as each dimension depends on the previously sampled dimensions.

Our specific implementation details, including the transformer architecture, attention mechanisms, and hyperparameters, can be found in Section 2.3 of Butter et al. (2023).

3.5.5 Results

Toy densities

We test the Bayesian versions of the models on two different two-dimensional toy datasets:

1. A ramp, linear in one dimension and flat in the other
2. A ring, where the radial distribution is Gaussian

We find that the two diffusion models have similar performance to each other and to a normalizing flow, providing smooth fits to the densities. The behavior, especially in terms of uncertainty, is consistent with what one would expect from fitting a simple parametric model to the density.

The autoregressive transformer yields more accurate results when predicting bin probabilities rather than Gaussian mixture parameters. Its results differ significantly from the other models, with uncertainty patterns that deviate from those typically obtained from a simple fit.

See Section 3 in Butter et al. (2023) for plots and more in-depth analysis.

LHC events

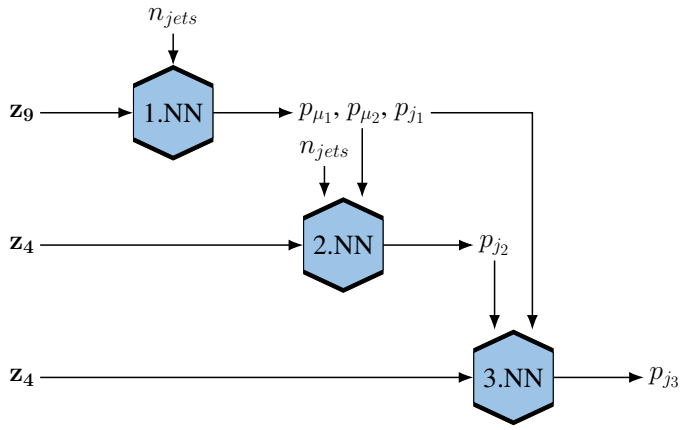


Figure 3.16: *Conditional sampling architecture.*

We benchmark our networks on a challenging set of LHC events, following the approach of Butter et al. (2021). The dataset consists of leptonically decaying Z bosons with a variable number of associated QCD jets: $pp \to Z_{\mu\mu} + \{1, 2, 3\}$ jets. Details of data generation and preprocessing are described in Section 4 of Butter et al. (2023).

The phase space dimensionality is 3 per muon (due to energy-momentum conservation) and 4 per jet. By exploiting azimuthal symmetry, we can remove one dimension, resulting in 9, 13, or 17 dimensions for 1, 2, or 3 jets, respectively. Jets are ordered by p_T.

Generating a variable number of jets poses a challenge for the DDPM and CFM models. Our solution, illustrated in Figure 3.16, involves a two-step process: first, we generate the 4-momenta of the two muons and the first jet (9 dimensions total), then we conditionally generate subsequent jet variables based on these initial values.

We compare our models to the normalizing flow presented in Figure 11 of Butter et al. (2021). In the subsequent figures, we display distributions of kinematic variables for different jet multiplicities. The most challenging correlations are shown in the right column:

- The Z peak, which requires learning a specific phase space direction with high precision.
- Jet-jet correlations, particularly the collinear enhancement at the hard jet-separation cut of $\Delta R_{jj} > 0.4$. This region of phase space is difficult to model due to its sharp boundary.

To assist the DDPM and CFM models, we employ a transformation described in Butter et al. (2021). This transformation makes the ΔR_{jj} features monotonically increasing, allowing us to learn the data in this transformed space before mapping the generated data back to the original space. Notably, the transformer model does not require this technique.

All models demonstrate proficiency in learning the p_T distributions.

Denoising diffusion The DDPM exhibits performance comparable to the normalizing flow in Butter et al. (2021), achieving precision predominantly at the 1% level, though occasionally reaching up to 10% (see Figure 3.17). However, the model's generation process is notably slow, requiring 1000 network calls per sample, with multiple sampling steps necessary for higher jet multiplicities.

Conditional flow matching The CFM model performs similarly to the DDPM but demonstrates superior modeling of the Z peak. The precision of the ΔR_{jj} distributions is comparable to that of the DDPM (see Figure 3.18). Notably, the CFM is approximately an order of magnitude faster due to its fairly linear trajectory, which is solved by an efficient ODE solver.

Autoregressive transformer The AT model distinguishes itself by generating events of all multiplicities within a single network. This is achieved by providing the multiplicity as an input and sampling the requisite number of variables for the given number of jets. The ordering of variables is crucial, as the model tends to learn early variables and correlations more effectively. Consequently, we prioritize the angles of the jets to ensure their accurate learning. While the Z peak is learned less accurately compared to the other models, the

ΔR_{jj} distributions are learned with approximately 1% precision without requiring the special transformation used for the other models (see Figure 3.19).

3.5.6 Summary

In this paper, we explore three Bayesian generative models for particle physics applications: denoising diffusion probabilistic models (DDPM), continuous flow matching (CFM), and autoregressive transformers (AT). These models are applied to two-dimensional toy datasets and a challenging set of LHC events involving leptonically decaying Z bosons with associated QCD jets ($pp \to Z_{\mu\mu} + \{1, 2, 3\}$ jets). The models are designed to generate high-dimensional phase space distributions, with the number of dimensions ranging from 9 to 17 depending on the number of jets.

Our results demonstrate that all three models perform well in learning complex phase space distributions, with each model showing distinct strengths. The DDPM and CFM models achieve precision predominantly at the 1% level, occasionally reaching up to 10%, and perform similarly to existing normalizing flow models. The CFM model, in particular, shows superior modeling of the Z peak and is approximately an order of magnitude faster than the DDPM. The AT model distinguishes itself by generating events of all multiplicities within a single network and learns the ΔR_{jj} distributions with approximately 1% precision without requiring special transformations. While each model has its strengths, they all demonstrate proficiency in learning p_T distributions and capturing complex phase space features, making them promising tools for particle physics simulations.

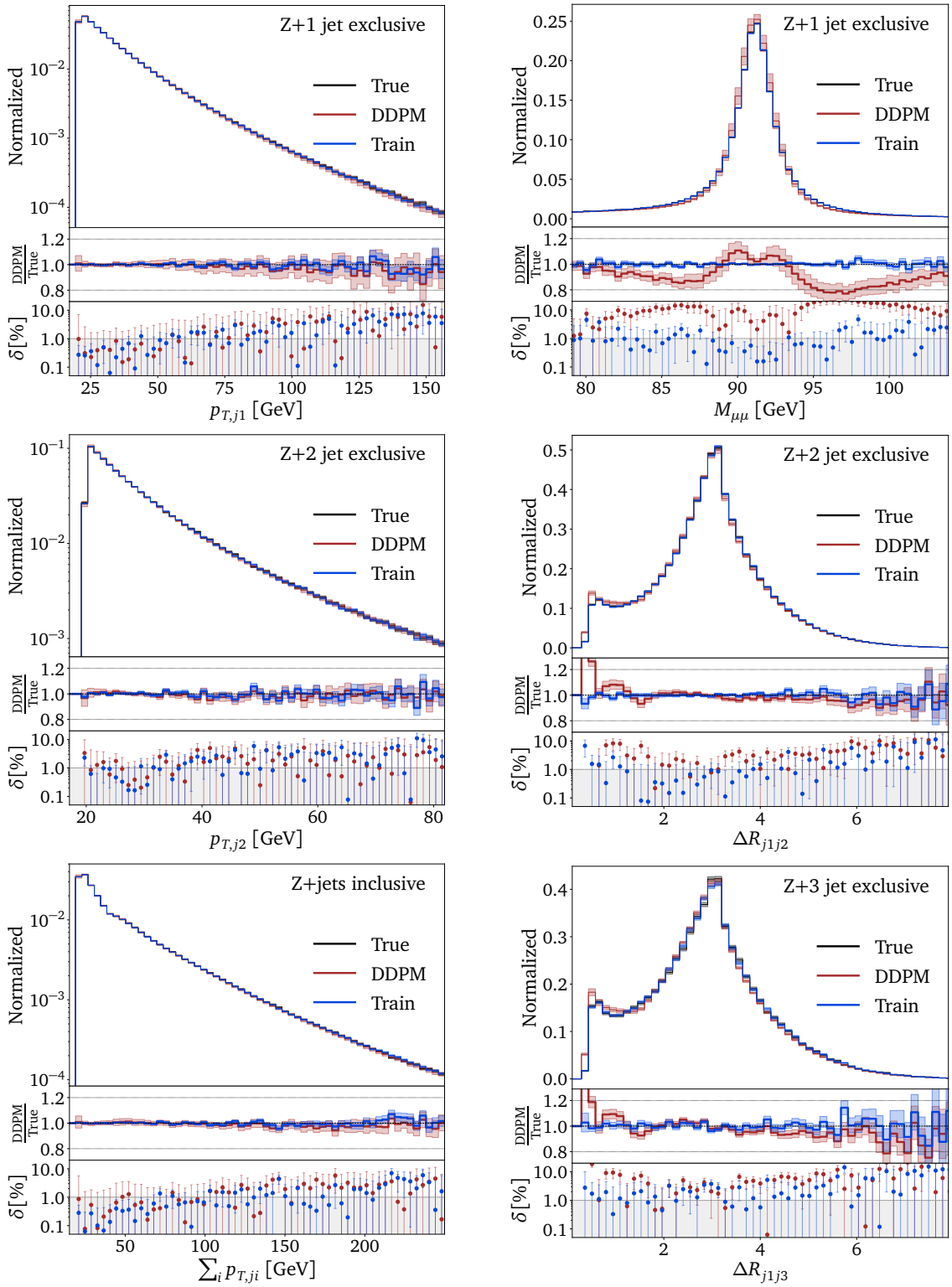


Figure 3.17: *Bayesian DDPM densities and uncertainties for $Z+1$ jet* (upper), *$Z+2$ jets* (center), *and $Z+3$ jets* (lower) *from combined $Z+$ jets generation. The uncertainty on the training data is given by bin-wise Poisson statistics. For a comparison with the INN we refer to Fig. 11 of Butter et al. (2021).*

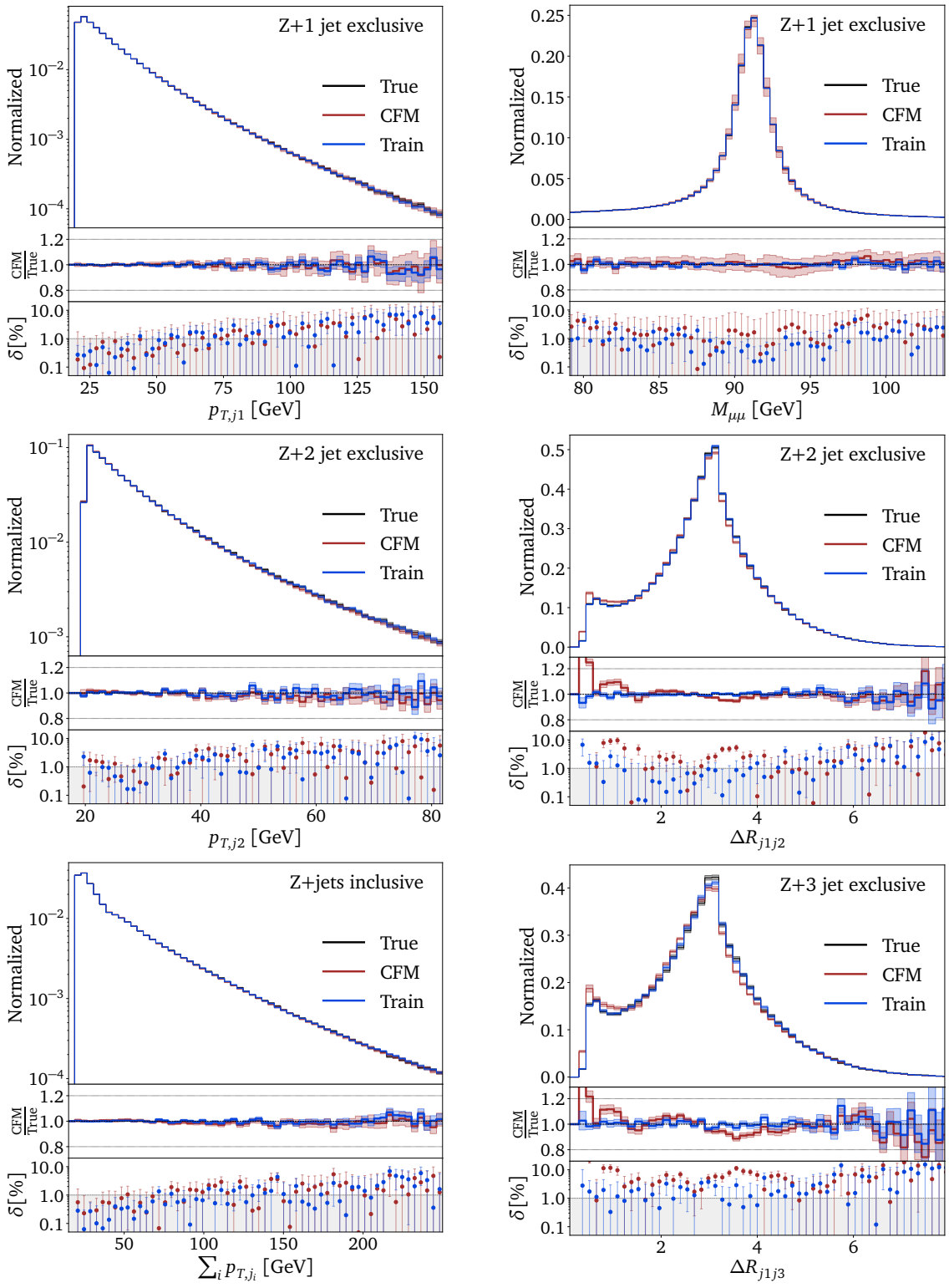


Figure 3.18: *Bayesian CFM densities and uncertainties for* $Z + 1$ *jet* (upper), $Z + 2$ *jets* (center), *and* $Z + 3$ *jets* (lower) *from combined* $Z+$ *jets generation. The uncertainty on the training data is given by bin-wise Poisson statistics. For a comparison with the INN we refer to Fig. 11 of Butter et al. (2021).*

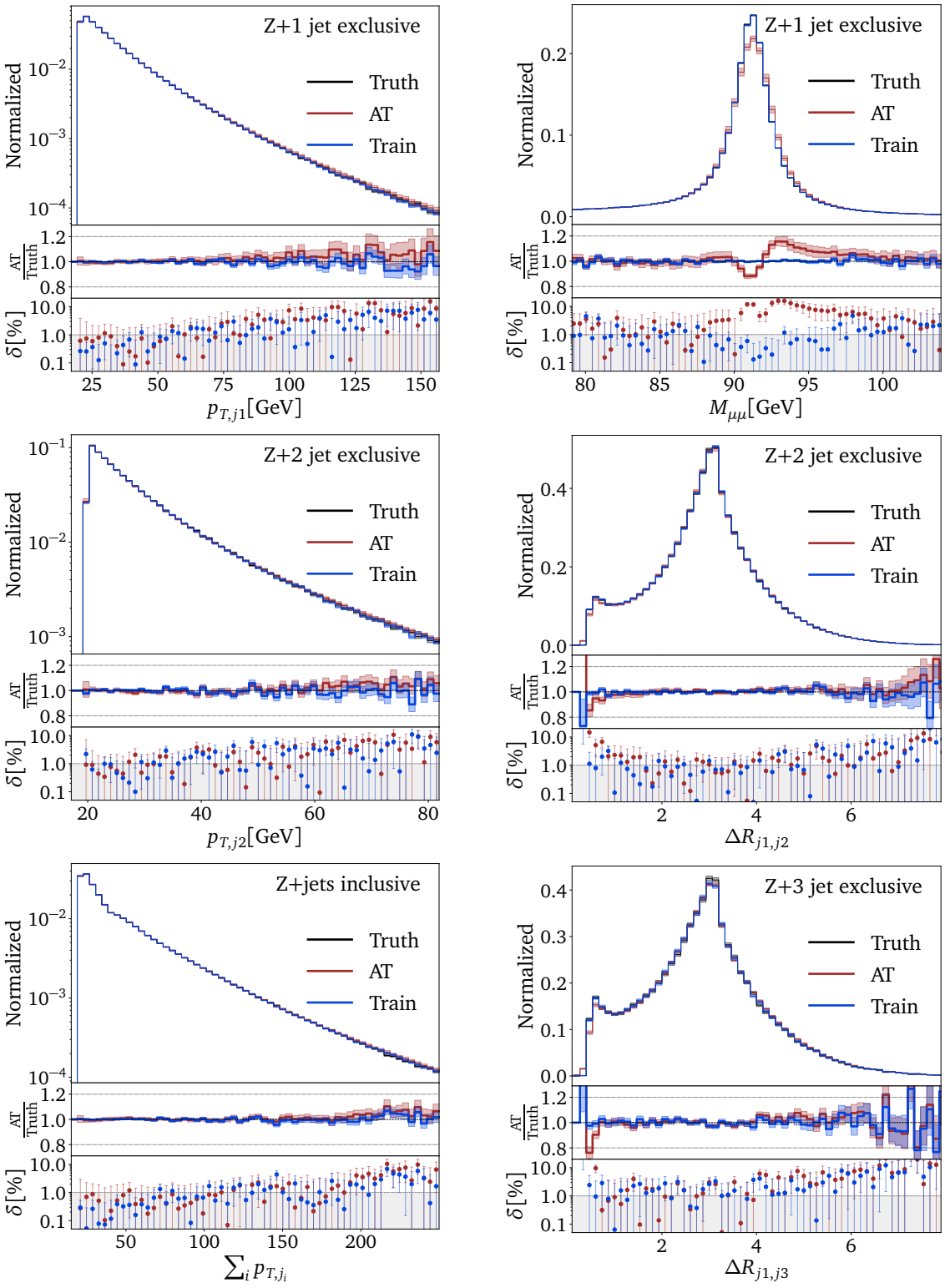


Figure 3.19: *Bayesian autoregressive transformer densities and uncertainties for $Z+1$ jet* (upper), *$Z+2$ jets* (center), *and $Z+3$ jets* (lower) *from combined $Z+$ jets generation. The uncertainty on the training data is given by bin-wise Poisson statistics. For a comparison with the INN we refer to Fig. 11 of Butter et al. (2021).*

Chapter 4

Conclusion

This thesis has explored two main areas of research: free-form flows and machine learning applications in particle physics. Both of these areas contribute to the advancement of generative modeling techniques, with a particular focus on scientific applications.

4.1 Free-form flows

Free-form flows fill an important niche in the field of generative modeling. They combine the strengths of normalizing flows—namely, exact maximum likelihood training and fast generation—with the flexibility of unrestricted neural network architectures. This approach addresses a key limitation of traditional normalizing flows: their restricted expressiveness due to architectural constraints.

Key contributions in this area include:

- The introduction of a novel maximum likelihood gradient estimator based on Jacobi's formula, which, combined with a reconstruction loss, allows for more flexible model architectures.
- The development of a theoretical framework for free-form flows, including proofs of their properties and their relationship to normalizing flows.
- The establishment of error bounds for the free-form flow estimator, providing theoretical guarantees for the approach.
- The demonstration of links between free-form flows and variational autoencoders, bridging two areas of generative modeling.
- The extension of free-form flows to bottleneck autoencoding models, making it possible to simultaneously learn a manifold and a probability distribution on it with a single optimization objective.
- The extension of free-form flows to Riemannian manifolds, allowing for generative modeling on non-Euclidean spaces such as spheres and tori.

These advancements open up new possibilities for designing generative models that can better capture the intricacies of complex scientific phenomena. The increased flexibility of free-form flows allows for the incorporation of domain-specific knowledge and the use

of tailored architectures, potentially leading to more accurate and interpretable models across various scientific disciplines.

4.1.1 Unifying view of free-form flows

The three free-form flow settings examined in this thesis—full-dimensional, bottleneck, and Riemannian manifolds—exhibit a shared maximum likelihood estimator with remarkably similar implementations. The maximum-likelihood component of the loss function can be consistently expressed as:

$$\mathcal{L}_{\mathrm{ML}} = \mathbb{E}_{x,v}[-\log p_Z(f(x)) - v^T \mathtt{SG}[g'(f(x))] f'(x) v] \tag{4.1}$$

This formulation has the potential to extend to other contexts, such as generating models of functions. In this case, f and g would act as functional operators, taking functions as inputs and producing functions as outputs, while x and v would represent appropriately sampled functions. This generalization could open new paths for modeling complex functional spaces across scientific domains. Applications could include modeling quantum wavefunctions in physics, gene expression patterns in biology, or time-varying climate models in environmental science, thus expanding the use of free-form flows to a broad range of scientific problems.

4.2 Machine learning in particle physics

The second major focus of this thesis has been the application of advanced machine learning techniques to problems in particle physics, particularly in the context of the Large Hadron Collider (LHC). These applications aim to accelerate the processing of vast data streams from the LHC and potentially uncover new physics beyond the Standard Model.

Key contributions in this area include:

- The introduction of a Dirichlet variational autoencoder (DVAE) for improved performance in distinguishing QCD background jets from top quark jets.
- The creation of JetCLR, a contrastive learning approach for jet representation that incorporates physical symmetries and augmentations, demonstrating superior performance in downstream top tagging tasks.
- The development of a normalized autoencoder (NAE) for LHC triggers, which upgrades a standard autoencoder to a probabilistic model, enabling symmetric identification of anomalous jets in both higher and lower complexity directions, at a low cost of inference.
- The exploration of generative models for LHC event simulation, including denoising diffusion probabilistic models (DDPM), continuous flow matching (CFM), and autoregressive transformers (AT).

These applications showcase the potential of advanced machine learning techniques to enhance our understanding of fundamental particle physics. By leveraging models with rich representational spaces, we can improve the efficiency and accuracy of data analysis at the LHC, potentially leading to new discoveries in the field.

4.3 Common threads

A common thread throughout this thesis is the development and application of generative models with increased flexibility and expressiveness. Whether in the context of free-form flows or particle physics applications, the focus has been on creating models that can better capture complex data distributions while incorporating domain-specific knowledge.

The methods presented share key characteristics:

- They rely on generative models based on maximum likelihood principles, whether exact maximum likelihood (as in autoregressive or energy-based models), approximately exact (as in free-form flows), or bounded (as in the variational autoencoders and diffusion models used in particle physics applications).
- They demonstrate the importance of incorporating domain-specific knowledge, whether through flexible architectures in free-form flows or through physics-inspired augmentations in JetCLR.

4.4 Future directions

The work presented in this thesis opens up several promising avenues for future research:

- Further exploration of the theoretical properties of free-form flows, including investigations into their stability and convergence characteristics.
- Application of free-form flows to a wider range of scientific problems, particularly in domains with complex, high-dimensional data.
- Extension of the free-form flow framework to more abstract generative problems, such as learning distributions over function spaces and probability distributions, enabling new applications in functional modeling and uncertainty quantification.
- Development of more sophisticated physics-inspired machine learning models for the LHC, potentially incorporating ideas from free-form flows.

In conclusion, this thesis has contributed to the advancement of generative modeling techniques with a focus on scientific applications. By developing more flexible and expressive models, we have opened up new possibilities for capturing complex phenomena in various scientific domains. The methods presented here, from free-form flows to advanced machine learning techniques in particle physics, represent steps towards more powerful and interpretable models for scientific data analysis and simulation. As these techniques continue to evolve, they hold the promise of accelerating scientific discovery and deepening our understanding of complex systems across a wide range of disciplines.

Bibliography

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *International Conference on Learning Representations*. 2023.

Anders Andreassen, Ilya Feige, Christopher Frye, and Matthew D. Schwartz. JUNIPR: A framework for unsupervised machine learning in particle physics. *The European Physical Journal*. 2019.

Keith A. Ashman, Christina M. Bird, and Stephen E. Zepf. Detecting bimodality in astronomical datasets. *The Astronomical Journal*. 1994.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. 2016.

Joshua Batson, C. Grace Haaf, Yonatan Kahn, and Daniel A. Roberts. Topological obstructions to autoencoding. *Journal of High Energy Physics*. 2021.

Hugues Beauchesne, Enrico Bertuzzo, Giovanni Grilli Di Cortona, and Zahra Tabrizi. Collider phenomenology of Hidden Valley mediators of spin 0 or 1/2 with semivisible jets. *Journal of High Energy Physics*. 2018.

Heli Ben-Hamu, Samuel Cohen, Joey Bose, Brandon Amos, Maximillian Nickel, Aditya Grover, Ricky TQ Chen, and Yaron Lipman. Matching normalizing flows and probability paths on manifolds. *International Conference on Machine Learning*. 2022.

Elias Bernreuther, Felix Kahlhoefer, Michael Krämer, and Patrick Tunney. Strongly interacting dark sectors in the early universe and at the LHC through a simplified portal. *Journal of High Energy Physics*. 2020.

Elias Bernreuther, Thorben Finke, Felix Kahlhoefer, Michael Krämer, and Alexander Mück. Casting a graph net to catch dark showers. *SciPost Physics*. 2021.

Andrew Blance, Michael Spannowsky, and Philip Waite. Adversarially-trained autoencoders for robust unsupervised new physics searches. *Journal of High Energy Physics*. 2019.

Alexander Bogatskiy, Brandon Anderson, Jan T. Offermann, Marwah Roussi, David W. Miller, and Risi Kondor. Lorentz group equivariant neural network for particle physics. 2020.

Joey Bose, Ariella Smofsky, Renjie Liao, Prakash Panangaden, and Will Hamilton. Latent variable modelling with hyperbolic normalizing flows. *International Conference on Machine Learning*. 2020.

Denis Boyda, Gurtej Kanwar, Sébastien Racanière, Danilo Jimenez Rezende, Michael S. Albergo, Kyle Cranmer, Daniel C. Hackett, and Phiala E. Shanahan. Sampling using $SU(N)$ gauge equivariant flows. *Physical Review D: Particles and Fields*. 2021.

G Brakenridge. Global active archive of large flood events. 2017.

Johann Brehmer and Kyle Cranmer. Flows for simultaneous manifold learning and density estimation. *Advances in Neural Information Processing Systems*. 2020.

James A Brofos, Marcus A Brubaker, and Roy R Lederman. Manifold density estimation via generalized dequantization. 2021.

Thorsten Buss, Barry M. Dillon, Thorben Finke, Michael Krämer, Alessandro Morandini, Alexander Mück, Ivan Oleksiyuk, and Tilman Plehn. What's anomalous in LHC jets? 2022.

Anja Butter and Tilman Plehn. Generative networks for LHC events. 2020.

Anja Butter, Gregor Kasieczka, Tilman Plehn, and Michael Russell. Deep-learned top tagging with a Lorentz layer. *SciPost Physics*. 2018.

Anja Butter, Theo Heimel, Sander Hummerich, Tobias Krebs, Tilman Plehn, Armand Rousselot, and Sophia Vent. Generative networks for precision enthusiasts. 2021.

Anja Butter, Tilman Plehn, Steffen Schumann, et al. Machine learning and LHC event generation. 2022.

Anja Butter, Nathan Huetsch, Sofia Palacios Schweitzer, Tilman Plehn, Peter Sorrenson, and Jonas Spinner. Jet Diffusion versus JetGPT – Modern Networks for the LHC. 2023.

Sascha Caron, Luc Hendriks, and Rob Verheyen. Rare and different: Anomaly scores from a combination of likelihood and out-of-distribution models to detect new physics at the LHC. 2021.

Stefano Carrazza and Frédéric A. Dreyer. Lund jet images from generative and cycle-consistent adversarial networks. *The European Physical Journal C: Particles and Fields*. 2019.

Anthony L Caterini, Gabriel Loaiza-Ganem, Geoff Pleiss, and John P Cunningham. Rectangular flows for manifold learning. *Advances in Neural Information Processing Systems*. 2021.

Clément Chadebec, Louis J. Vincent, and Stéphanie Allassonnière. Pythae: Unifying generative autoencoders in Python – a benchmarking use case. *Advances in Neural Information Processing Systems*. 2022.

Amit Chakraborty, Sung Hak Lim, and Mihoko M. Nojiri. Interpretable deep learning for two-prong jet classification with jet spectra. *Journal of High Energy Physics*. 2019.

Ricky TQ Chen and Yaron Lipman. Flow matching on general geometries. *International Conference on Learning Representations*. 2024.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*. 2018.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Everest Hinton. A simple framework for contrastive learning of visual representations. 2020.

Josh Cogan, Michael Kagan, Emanuel Strauss, and Ariel Schwarztman. Jet-images: Computer vision inspired techniques for jet tagging. *Journal of High Energy Physics*. 2015.

Timothy Cohen, Mariangela Lisanti, and Hou Keong Lou. Semivisible jets: Dark matter undercover at the LHC. *Physical Review Letters*. 2015.

Timothy Cohen, Mariangela Lisanti, Hou Keong Lou, and Siddharth Mishra-Sharma. LHC searches for dark sector showers. *Journal of High Energy Physics*. 2017.

Valentin De Bortoli, Emile Mathieu, Michael Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modelling. *Advances in Neural Information Processing Systems*. 2022.

Luke de Oliveira, Michael Kagan, Lester Mackey, Benjamin Nachman, and Ariel Schwartzman. Jet-images — deep learning edition. *Journal of High Energy Physics*. 2016.

B. M. Dillon, D. A. Faroughy, J. F. Kamenik, and M. Szewc. Learning the latent structure of collider events. *Journal of High Energy Physics*. 2020.

Barry M. Dillon, Darius A. Faroughy, and Jernej F. Kamenik. Uncovering latent jet substructure. *Physical Review D: Particles and Fields*. 2019.

Barry M. Dillon, Tilman Plehn, Christof Sauer, and Peter Sorrenson. Better latent spaces for better autoencoders. *SciPost Physics*. 2021.

Barry M. Dillon, Gregor Kasieczka, Hans Olischlager, Tilman Plehn, Peter Sorrenson, and Lorenz Vogel. Symmetries, safety, and self-supervision. *SciPost Physics*. 2022.

Barry M. Dillon, Luigi Favaro, Michael Krämer, Tilman Plehn, and Peter Sorrenson. A normalized autoencoder for LHC triggers. *SciPost Physics Core*. 2023.

Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with Gaussian mixture variational autoencoders. 2016.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation. *International Conference on Learning Representations, Workshop Track*. 2015.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *International Conference on Learning Representations*. 2017.

Tommaso Dorigo, Martina Fumanelli, Chiara Maccani, Marija Mojsovska, Giles C. Strong, and Bruno Scarpa. RanBox: Anomaly detection in the copula space. 2021.

Felix Draxler, Peter Sorrenson, Lea Zimmermann, Armand Rousselot, and Ullrich Köthe. Free-form flows: Make any architecture a normalizing flow. *International Conference on Artificial Intelligence and Statistics*. 2024.

Frédéric A. Dreyer, Gavin P. Salam, and Grégory Soyez. The Lund jet plane. *Journal of High Energy Physics*. 2018.

Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. 2019.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *Advances in Neural Information Processing Systems*. 2019.

EOSDIS. Land, atmosphere near real-time capability for EOS (LANCE) system operated by NASA's earth science data and information system (ESDIS). 2020.

M. Erdmann, E. Geiser, Y. Rath, and M. Rieger. Lorentz boost networks: Autonomous physics-inspired feature engineering. *Journal of Instrumentation*. 2019.

Luca Falorsi. Continuous normalizing flows on manifolds. 2021.

Luca Falorsi and Patrick Forré. Neural ordinary differential equations on manifolds. 2020.

Luca Falorsi, Pim de Haan, Tim R Davidson, and Patrick Forré. Reparameterizing distributions on Lie groups. *International Conference on Artificial Intelligence and Statistics*. 2019.

Marco Farina, Yuichiro Nakai, and David Shih. Searching for new physics with deep autoencoders. *Physical Review D: Particles and Fields*. 2020.

Thorben Finke. Deep learning for new physics searches at the LHC. 2020.

Thorben Finke, Michael Krämer, Alessandro Morandini, Alexander Mück, and Ivan Oleksiyuk. Autoencoders for unsupervised anomaly detection in high energy physics. *Journal of High Energy Physics*. 2021.

Katherine Fraser, Samuel Homiller, Rashmish K. Mishra, Bryan Ostdiek, and Matthew D. Schwartz. Challenges for unsupervised anomaly detection in particle physics. 2021.

Jerome H Friedman. On multivariate goodness–of–fit and two–sample testing. *Statistical Problems in Particle Physics, Astrophysics, and Cosmology*. 2003.

Mevlana C Gemici, Danilo Rezende, and Shakir Mohamed. Normalizing flows on Riemannian manifolds. 2016.

A Girard. A fast 'Monte-Carlo cross-validation' procedure for large least squares problems with noisy data. *Numerische Mathematik*. 1989.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations*. 2019.

Luigi Gresele, Giancarlo Fissore, Adrián Javaloy, Bernhard Schölkopf, and Aapo Hyvarinen. Relative gradient optimization of the Jacobian term in unsupervised deep learning. *Advances in Neural Information Processing Systems*. 2020.

Yifan Guo, Weixian Liao, Qianlong Wang, Lixing Yu, Tianxi Ji, and Pan Li. Multidimensional time series anomaly detection: A GRU-based Gaussian mixture variational autoencoder approach. *Asian Conference on Machine Learning*. 2018.

Theo Heimel, Gregor Kasieczka, Tilman Plehn, and Jennifer M. Thompson. QCD or what? *SciPost Physics*. 2019.

Isaac Henrion, Kyle Cranmer, Joan Bruna, Kyunghyun Cho, Johann Brehmer, Gilles Louppe, and Gaspar Rochette. Neural message passing for jet physics. *Proceedings of the Deep Learning for Physical Sciences Workshop at NIPS*. 2017.

John R Hershey and Peder A Olsen. Approximating the Kullback Leibler divergence between Gaussian mixture models. *IEEE International Conference on Acoustics, Speech and Signal Processing*. 2007.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. *Advances in Neural Information Processing Systems*. 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*. 2020.

Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3D. *International Conference on Machine Learning*. 2022.

Christian Horvat and Jean-Pascal Pfister. Denoising normalizing flow. *Advances in Neural Information Processing Systems*. 2021.

Chin-Wei Huang, Milad Aghajohari, Joey Bose, Prakash Panangaden, and Aaron C Courville. Riemannian diffusion models. *Advances in Neural Information Processing Systems*. 2022.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*. 1989.

Weonyoung Joo, Wonsung Lee, Sungrae Park, and Il-Chul Moon. Dirichlet variational autoencoder. *Pattern Recognition*. 2020.

Jürgen Jost. *Riemannian Geometry and Geometric Analysis*. 2008.

Xiangyang Ju and Benjamin Nachman. Supervised jet clustering with graph neural networks for Lorentz boosted bosons. 2020.

Dimitris Kalatzis, Johan Ziruo Ye, Alison Pouplin, Jesper Wohlert, and Søren Hauberg. Density estimation on smooth manifolds with normalizing flows. 2021.

Gurtej Kanwar, Michael S. Albergo, Denis Boyda, Kyle Cranmer, Daniel C. Hackett, Sébastien Racanière, Danilo Jimenez Rezende, and Phiala E. Shanahan. Equivariant flow-based sampling for lattice gauge theory. *Physical Review Letters*. 2020.

G. Kasieczka, T. Plehn, A. Butter, K. Cranmer, D. Debnath, B. M. Dillon, M. Fairbairn, D. A. Faroughy, W. Fedorko, C. Gay, L. Gouskos, J. F. Kamenik, P. T. Komiske, S. Leiss, A. Lister, S. Macaluso, E. M. Metodiev, L. Moore, B. Nachman, K. Nordstrom, J. Pearkes, H. Qu, Y. Rath, M. Rieger, D. Shih, J. M. Thompson, and S. Varma. The machine learning landscape of top taggers. *SciPost Physics*. 2019.

Gregor Kasieczka, Tilman Plehn, Michael Russell, and Torben Schell. Deep-learning top taggers or the end of QCD? *Journal of High Energy Physics*. 2017.

Gregor Kasieczka et al. The LHC Olympics 2020: A community challenge for anomaly detection in high energy physics. *Reports on Progress in Physics*. 2021.

Thomas A Keller, Jorn WT Peters, Priyank Jaini, Emiel Hoogeboom, Patrick Forré, and Max Welling. Self normalizing flows. *International Conference on Machine Learning*. 2021.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *International Conference on Learning Representations*. 2014.

Leon Klein, Andreas Krämer, and Frank Noé. Equivariant flow matching. *Advances in Neural Information Processing Systems*. 2023.

Simon Knapen, Jessie Shelton, and Dong Xu. Perturbative benchmark models for a dark shower search program. *Physical Review D: Particles and Fields*. 2021.

Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2021.

Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: Exact likelihood generative learning for symmetric densities. *International Conference on Machine Learning*. 2020.

Patrick T. Komiske, Eric M. Metodiev, and Matthew D. Schwartz. Deep learning in color: Towards automated quark/gluon jet discrimination. *Journal of High Energy Physics*. 2017.

Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler. Energy flow polynomials: A complete linear basis for jet substructure. *Journal of High Energy Physics*. 2018.

Konik Kothari, AmirEhsan Khorashadizadeh, Maarten V. de Hoop, and Ivan Dokmanic. Trumpets: Injective flows for inference and inverse problems. *Conference on Uncertainty in Artificial Intelligence*. 2021.

Steven G Krantz and Harold R Parks. *Geometric Integration Theory*. 2008.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Jim Lawrence, Javier Bernal, and Christoph Witzgall. A purely algebraic justification of the Kabsch-Umeyama algorithm. *Journal of research of the National Institute of Standards and Technology*. 2019.

Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. 2010.

Joshua Lin, Marat Freytsis, Ian Moult, and Benjamin Nachman. Boosting $H \to b\bar{b}$ with machine learning. *Journal of High Energy Physics*. 2018.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *International Conference on Learning Representations*. 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. 2022.

Yulin Liu, Haoran Liu, Yingda Yin, Yang Wang, Baoquan Chen, and He Wang. Delving into discrete normalizing flows on $SO(3)$ manifold for probabilistic rotation modeling. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. *Proceedings of International Conference on Computer Vision (ICCV)*. December 2015.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. *International Conference on Learning Representations*. 2017.

Aaron Lou, Derek Lim, Isay Katsman, Leo Huang, Qingxuan Jiang, Ser Nam Lim, and Christopher M De Sa. Neural manifold ordinary differential equations. *Advances in Neural Information Processing Systems*. 2020.

Gilles Louppe, Kyunghyun Cho, Cyril Becot, and Kyle Cranmer. QCD-aware recursive neural networks for jet physics. *Journal of High Energy Physics*. 2019.

Simon C. Lovell, Ian W. Davis, W. Bryan Arendall III, Paul I. W. de Bakker, J. Michael Word, Michael G. Prisant, Jane S. Richardson, and David C. Richardson. Structure validation by Cα geometry: ϕ, ψ and Cβ deviation. *Proteins: Structure, Function, and Bioinformatics*. 2003.

Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking simulation-based inference. *International Conference on Artificial Intelligence and Statistics*. 2021.

Sebastian Macaluso and David Shih. Pulling out all the tops with computer vision and deep learning. *Journal of High Energy Physics*. 2018.

Jan R. Magnus and Heinz Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. 2019.

Emile Mathieu and Maximilian Nickel. Riemannian continuous normalizing flows. *Advances in Neural Information Processing Systems*. 2020.

David E. Morrissey, Tilman Plehn, and Tim M. P. Tait. Physics searches at the LHC. *Physics Reports*. 2012.

Kieran A Murphy, Carlos Esteves, Varun Jampani, Srikumar Ramalingam, and Ameesh Makadia. Implicit-PDF: Non-parametric representation of probability distributions on the rotation manifold. *International Conference on Machine Learning*. 2021.

Laura Murray, W Arendall, David Richardson, and Jane Richardson. RNA backbone is rotameric. *Proceedings of the National Academy of Sciences of the United States of America*. 2003.

Benjamin Nachman and David Shih. Anomaly detection with density estimation. *Physical Review D: Particles and Fields*. 2020.

National Geophysical Data Center / World Data Service NGDC/WDS. NCEI/WDS global significant earthquake database. 2022a.

National Geophysical Data Center / World Data Service NGDC/WDS. NCEI/WDS global significant volcanic eruptions database. 2022b.

Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*. 2019.

Ivan Oleksiyuk. Unsupervised learning for tagging anomalous jets at the LHC. 2020.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in Neural Information Processing Systems*. 2017.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*. 2021.

Athanasios Papoulis and S. Unnikrishna Pillai. *Probability, Random Variables and Stochastic Processes*. 2002.

David Peel, William J Whiten, and Geoffrey J McLachlan. Fitting mixtures of Kent distributions to aid in joint set identification. *Journal of the American Statistical Association*. 2001.

Aaron Pierce, Bibhushan Shakya, Yuhsin Tsai, and Yue Zhao. Searching for confining hidden valleys at LHCb, ATLAS, and CMS. *Physical Review D: Particles and Fields*. 2018.

Shah Rukh Qasim, Jan Kieseler, Yutaro Iiyama, and Maurizio Pierini. Learning representations of irregular particle-detector geometry with distance-weighted graph networks. *The European Physical Journal C: Particles and Fields*. 2019.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*. 2019.

G. N. Ramachandran, C. Ramakrishnan, and V. Sasisekharan. Stereochemistry of polypeptide chain configurations. *Journal of Molecular Biology*. 1963.

R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*. 2014.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. *International Conference on Machine Learning*. 2015.

Danilo Jimenez Rezende, George Papamakarios, Sébastien Racaniere, Michael Albergo, Gurtej Kanwar, Phiala Shanahan, and Kyle Cranmer. Normalizing flows on tori and spheres. *International Conference on Machine Learning*. 2020.

Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*. 1996.

Tuhin S. Roy and Aravind H. Vijay. A robust anomaly finder based on autoencoders. 2019.

Noam Rozen, Aditya Grover, Maximilian Nickel, and Yaron Lipman. Moser flow: Divergence-based generative modeling on manifolds. *Advances in Neural Information Processing Systems*. 2021.

L. Ruddigkeit, R. van Deursen, L. C. Blum, and J.-L. Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. *Journal of Chemical Information and Modeling*. 2012.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. *Advances in Neural Information Processing Systems*. 2016.

Victor Garcia Satorras, Emiel Hoogeboom, Fabian Fuchs, Ingmar Posner, and Max Welling. E(n) equivariant normalizing flows. *Advances in Neural Information Processing Systems*. 2021.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. *International Conference on Machine Learning*. 2021b.

Jie Shao and Xiaorui Li. Generalized zero-shot learning with multi-channel Gaussian mixture VAE. *IEEE Signal Processing Letters*. 2020.

Chase Shimmin. Particle convolution for high energy physics. 2021.

Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. 2020.

Rui Shu, James Brofos, Frank Zhang, Hung Hai Bui, Mohammad Ghavamzadeh, and Mykel Kochenderfer. Stochastic video prediction with conditional density estimation. *ECCV Workshop on Action and Anticipation for Visual Learning*. 2016.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *International Conference on Machine Learning*. 2015.

Yang Song and Diederik P. Kingma. How to train your energy-based models. 2021.

Peter Sorrenson, Felix Draxler, Armand Rousselot, Sander Hummerich, and Ullrich Köthe. Learning distributions on manifolds with free-form flows. *Advances in Neural Information Processing Systems*. 2024a.

Peter Sorrenson, Felix Draxler, Armand Rousselot, Sander Hummerich, Lea Zimmermann, and Ullrich Köthe. Lifting architectural constraints of injective flows. *International Conference on Learning Representations*. 2024b.

Akash Srivastava and Charles Sutton. Autoencoding variational inference for topic models. *ICLR*. 2017.

Matthew J. Strassler and Kathryn M. Zurek. Echoes of a hidden valley at hadron colliders. *Physics Letters B*. 2007.

Yunfei Teng and Anna Choromanska. Invertible autoencoder for domain adaptation. *Computation*. 2019.

Jakub Tomczak and Max Welling. VAE with a VampPrior. *International Conference on Artificial Intelligence and Statistics*. 2018.

Greg Turk and Marc Levoy. Zippered polygon meshes from range images. *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. 1994.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. July 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*. 2017.

Jonas Wildberger, Maximilian Dax, Simon Buchholz, Stephen R Green, Jakob H Macke, and Bernhard Schölkopf. Flow matching for scalable simulation-based inference. *Advances in Neural Information Processing Systems*. 2023.

Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. A theory of generative ConvNet. 2016.

Linxiao Yang, Ngai-Man Cheung, Jiaying Li, and Jun Fang. Deep clustering by Gaussian mixture variational autoencoders with graph embedding. *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

Sangwoong Yoon, Yung-Kyun Noh, and Frank Chongwoo Park. Autoencoding under normalization constraints. *International Conference on Machine Learning*. 2021.

Mingtian Zhang, Peter Hayes, Thomas Bird, Raza Habib, and David Barber. Spread divergence. *International Conference on Machine Learning*. 2020.