Scalable and Flexible Density Estimation for Complex Data Distributions

Etrit Haxholli

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

Thèse de Doctorat

Estimation de Densité Flexible et Efficace pour les Distributions des Données Complexes

Etrit Haxholli

Inria, Équipe Epione

Thèse dirigée par Marco Lorenzi

Soutenue le 25 octobre 2023

Présentée en vue de l'obtention du grade de Docteur en Automatique, Traitement du Signal et des Images de l'Université Côte d'Azur.

Devant le jury composé de :

Bertrand Thirion	Professeur à l'Université Paris-Saclay	Rapporteur
Marc Niethammer	Professeur à l'Université de Caroline du Nord à Chapel Hill	Rapporteur
Pietro Michiardi	Professeur à Eurecom Sophia Antipolis	Examinateur
Marco Lorenzi	Chargé de Recherche au Center INRIA d'Université Côte d'Azur	Directeur de thèse

Abstract

Density estimation is the statistical process of constructing a probabilistic model that represents the distribution of a given dataset. By estimating this distribution, we can better understand the statistics and behavior of our data, enhancing predictions, anomaly detection, and data generation. Density estimation thus forms a crucial step in numerous data analysis and machine learning tasks. Nonetheless, the task of modeling high-dimensional distributions introduces a multitude of challenges. These primarily arise from the need to develop models that exhibit flexibility, which allows for the precise capture of complex high-dimensional patterns, and computational feasibility that becomes particularly essential during the training phase. Within this context, the thesis aims to contribute new theoretical and practical perspectives specifically designed to refine the modeling of probability density functions for high-dimensional, complex data distributions. Furthermore, we propose an innovative theory to improve the quantification of the properties of distributions, such as the thickness of their tails, under statistical assumptions relevant to the machine learning setting.

The initial part of this thesis focuses on the examination of deep neural function approximators that are assured to represent probability density functions, regardless of the parameters' values. To achieve this, we introduce AFFJORD, an enhanced version of Continuous Normalizing Flows. This enhancement is made possible through augmentation, inspired by our derivation of the Jacobian of diffeomorphic transformations parameterized by Ordinary Differential Equations (ODEs). Additionally, we expand on the recent advancements in diffusion models, proposing a novel method (PSM) that enhances density estimation while accelerating training, without incurring any drawbacks in terms of inference time or memory consumption. This is achieved by exploiting the independence inherent in modeling the scores at different time points within diffusion models. The result is a flexible, rapidly optimizable, piecewise continuous normalizing flow.

In the second part of the thesis, we illustrate that the finiteness of the sampling procedure from marginal distributions negatively impacts the reliability and efficiency of traditional tail estimation methods derived from Extreme Value Theory, such as the Peaks-Over-

Threshold approach. To address this challenge, we devise an innovative general theory for estimating the tails of marginal distributions, particularly when there is significant variability between locations of the individual conditional distributions that underlie the marginal. Under certain regularity conditions, we demonstrate that the shape parameter of the marginal distribution corresponds to the maximum tail shape parameter of the family of conditional distributions. This estimation approach is coined as 'cross-tail estimation (CTE)'. We validate cross-tail estimation via a series of experiments conducted on both simulated and real data. Our findings showcase the improved robustness and superior quality of tail estimation in comparison to conventional methodologies and provide evidence of the correlation between overfitting and the thickness of the loss distribution tail.

In conclusion, this work offers novel theoretical insights and practical perspectives, specifically tailored to enhance the modeling of probability density functions for high-dimensional, intricate data distributions. Additionally, we propose an innovative theory aimed at refining the quantification of the tail thickness of distributions, under statistical assumptions pertinent to the machine learning context. We conclude this thesis by outlining three potential avenues for future research within this field, which are then followed by our final remarks.

Keywords: Density Estimation, Normalizing Flows, Diffusion Models, Extreme Value Theory, Tail Modelling, Loss Function Distributions, Peaks-Over-Threshold, Cross-Tail-Estimation.

Résumé

L'estimation de la densité est le processus statistique de construction d'un modèle probabiliste qui représente la distribution d'un ensemble de données. En estimant cette distribution, nous pouvons mieux comprendre les statistiques et le proprietés des données, améliorant les prédictions, la détection d'anomalies, et la génération. L'estimation de la densité est donc une étape cruciale dans nombreuses tâches d'analyse de données et d'apprentissage. Néanmoins, la tâche de modélisation de distributions à haute dimension introduit de nombreux défis. Ces défis proviennent principalement du besoin de développer des modèles qui présentent une flexibilité, permettant la capture précise de motifs complexes à haute dimension, et d'une faisabilité computationnelle qui devient particulièrement essentielle durant la phase d'entraînement. Dans ce contexte, la thèse vise à apporter de nouvelles perspectives théoriques et pratiques, spécialement conçues pour affiner la modélisation des fonctions de densité de probabilité pour des distributions de données complexes à haute dimension. De plus, nous proposons une théorie innovante pour améliorer la quantification des propriétés des distributions, comme l'ampleurs de leurs queues.

La première partie de cette thèse se concentre sur l'etude des approximations de reseaux de neurones profondes concues pour représenter des fonctions de densité de probabilité, indépendamment des valeurs des paramètres. Pour cela, nous introduisons AFFJORD, comme extensions de l'etat de l'art sur les "normalizing flows". Cette amélioration est rendue possible grâce à une augmentation, inspirée par notre dérivation du jacobien des transformations diffeomorphiques paramétrées par des équations différentielles ordinaires (ODE). De plus, nous proposons une nouvelle méthode en s'appuyant sur les modeles de diffusion (PSM) qui améliore l'estimation de la densité tout en accélérant le processus d'entraiment, sans encourir d'inconvénients en termes de temps d'inférence ou de consommation de mémoire. Ceci est réalisé en exploitant l'indépendance inhérente à la modélisation des scores dans les modèles de diffusion. Le résultat est un "normalizing flow" continu par morceaux, flexible et rapidement optimisable.

La deuxième partie de la thèse illustre que la procédure d'échantillonnage à partir de distributions marginales a un impact négatif sur la fiabilité et l'efficacité des méthodes traditionnelles d'estimation de queue dérivées de la théorie des valeurs extrêmes. Pour

relever ce défi, nous développons une théorie générale innovante pour estimer les queues de distributions marginales, en particulier lorsque la variabilité est significative entre les distributions conditionnelles individuelles. Sous certaines conditions de régularité, nous démontrons que le paramètre de forme de la distribution marginale correspond au paramètre de forme de queue maximum de la famille de distributions conditionnelles.

En conclusion, ce travail offre de nouvelles perspectives théoriques et pratiques, spécialement conçues pour améliorer la modélisation des fonctions de densité de probabilité pour des distributions de données complexes à haute dimension. De plus, nous proposons une théorie innovante pour affiner la quantification de l'épaisseur des queues des distributions, en vertu d'hypothèses statistiques pertinentes dans le domaine de l'apprentissage statistique. Nous concluons cette thèse en proposant trois voies potentielles pour les recherches futures dans ce domaine. Celles-ci sont ensuite suivies par nos remarques finales.

Mots clés: Estimation de la Densité, Flux de Normalisation, Modèles de Diffusion, Théorie des Valeurs Extrêmes, Modélisation de la Queue, Distributions de la Fonction de Perte, Pics-Au-Dessus-du-Seuil, Estimation de la Queue Croisée.

Acknowledgements

I am deeply grateful to my supervisor Marco Lorenzi for the opportunity to pursue my thesis within the Epione team at Inria. His trust and confidence in me have been invaluable throughout this project. I sincerely appreciate Marco's guidance, advice, and the insightful discussions we have had. His passion, motivation, and expertise have been, and will continue to be a great source of inspiration to me. In addition, I wish to extend my sincere gratitude to Carlo Fanara, whose support and confidence in me were instrumental during the internship phase of this project. Working alongside him has been both a privilege and a source of pride. I would also like to thank Nicholas Ayache for welcoming me into the Epione team, for giving me the opportunity to organize the 3IA Seminar Series, and for the trust he placed in me.

Furthermore, I extend my gratitude to the jury members, Bertrand Thirion and Marc Niethammer, for their commitment in thoroughly reviewing my thesis, attending the defense, and offering insightful comments and advice. Additionally, my most sincere thanks go to Pietro Michiardi for his engaging and thought-provoking questions during the defense, which greatly contributed to the depth and quality of the discourse.

Throughout my doctoral journey, I have had the privilege of forming enriching relationships that have significantly enhanced my experience. The Epione team, comprised of individuals who are kind, thoughtful, helpful, modest, and intelligent, has been a cornerstone of both my PhD journey and my personal growth. The opportunity to work alongside such a distinguished group has undoubtedly been a highlight of my academic life. I am particularly grateful to Yann Fraboni, whose engaging and inspiring conversations have broadened my scientific perspectives and fostered personal development in numerous areas. In the same breath, my heartfelt thanks go to Dimitri Hamzaoui, whose friendship and support have been invaluable. In addition, I extend my appreciation to Santiago Silva, Luis Gomes Pereira, Riccardo Taiello, Lisa Guzzi, Lucia Innocenti, Martin Van Waerebeke, Jairo Rodriguez, Elodie Maignant, Zhijie Fang, James Benn, Hava Chaptoukaev, Josquin Harrison, Andrea Senacheribbe, Hari Sreedhar, Buntheng Ly and many others. Their collective influence has profoundly shaped my experience, for which I am deeply thankful.

Finally, I would like to thank my family for their unwavering support at every stage of my journey—preceding, throughout, and beyond the tenure of my PhD. Their presence

in both challenging and prosperous times has been an invaluable constant in my life. I am deeply appreciative of the fortune of having such a supportive family, whose encouragement and belief in my endeavors have been integral to my well-being. Thank you!

Financial Support

This work has been supported by the French government, through the 3IA Côte d'Azur Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002. The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.

Contents

1 Introduction

Contents

The objective of this inaugural chapter is to describe the context and objectives that guide this thesis. We commence by elaborating on the concept of density estimation, followed by a discussion on both traditional and contemporary methodologies that are deployed for the aforementioned task. Subsequently, a subsection is dedicated to the field of Extreme Value Theory, focusing specifically on the various techniques for estimating the shape of the tails of probability distributions.

We conclude this chapter by exploring some important challenges related to the methodologies discussed in density estimation and the estimation of the tail shape of marginal distributions, and by providing a summary of the steps taken in this thesis to mitigate the aforementioned challenges.

1.1 Probability Density Estimation

Density estimation is the statistical process of constructing a probabilistic model that represents the distribution of a given dataset. By estimating this distribution, we can better understand the statistics and behavior of our data, enhancing predictions, anomaly detection, and data generation. Density estimation thus forms a crucial step in numerous data analysis and machine learning tasks. Various methods have been developed for density estimation, encompassing both parametric and non-parametric approaches. In what

follows, we illustrate some of these methods, detailing the advantages and drawbacks inherent to each.

1.1.1 Classical Methods

Parametric Methods:

Parametric density estimation methods assume that the underlying probability density function belongs to a specific parametric family, whose parameters can be estimated using the observed data. Parametric methods for density estimation offer several advantages. The parameters of such models often have clear interpretations, allowing insights into the underlying distribution. Parametric methods can be computationally efficient, since the number of parameters can be fixed regardless of the sample size. However, they also have limitations, particularly when the assumptions of the chosen parametric family do not align well with the underlying data distribution. Furthermore, they may suffer from the curse of dimensionality, an issue that this thesis aims to address. Some prominent parametric methods include:

Families of basic probability density functions. In these models, we postulate that the data distribution belongs to a particular family of probability density functions $f(\boldsymbol{x}, \boldsymbol{\theta})$, such as the Gaussian, Beta, Gamma distributions among others [Casella, 2001]. Typically, parameters maximizing the likelihood of the observed data, given a specific parametric density function, are estimated either through analytical/numerical maximum likelihood estimation (MLE), or methods of moments. Regrettably, complex data distributions are generally challenging to be described by these standard families.

Mixtures of families of parametric probability density functions. Mixture models [Everitt, 1981] exploit the property that a convex combination of probability density functions is still a valid probability density function:

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{k=1}^{n} p_k f_k(\boldsymbol{x}, \boldsymbol{\theta}_k), \text{ where } \sum_{k=1}^{n} p_k = 1. \tag{1.1}$$

Under certain conditions, these mixtures, if composed of particular families of pdfs, can act as universal approximators—meaning they can theoretically approximate any given distribution to an arbitrary degree of accuracy, given sufficient components in the mixture [Goodfellow, 2016]. However, these models do not come without challenges. For instance, a Gaussian mixture model, $f_k(\boldsymbol{x}, \boldsymbol{\theta}_k) = p_k(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$, can collapse into singularities when optimized using methods such as Maximum Likelihood Estimation [Bishop, 2007; Shireman, 2015]. Moreover, when other methods like the Expectation-Maximization algorithm are utilized, the estimated parameters become highly sensitive to their initialization [Biernacki, 2003; Seidel, 2000].

On top of these, mixture models are not immune to the curse of dimensionality. In instances of high-dimensional and complex data distributions, the number of components needed for effective density estimation can become significantly large. Importantly, each component inherently contains a number of parameters that scales with the dimensionality of the data. The task of selecting the optimal number of components thus represents a considerable and non-trivial challenge.

Probabilistic Chain Rule (Autoregressive Models). While initially developed for modelling time series [Box, 1994] such models can be used for general density estimation tasks [Bengio, 1999; Larochelle, 2011; Uria, 2013]. They operate based on the principle of the chain rule of probability, expressed as:

$$p(\boldsymbol{x}) = p(x_1, ..., x_{d-1}, x_d) = p(x_d|x_{d-1}, ..., x_1) \dots p(x_2|x_1)p(x_1). \tag{1.2}$$

This formulation ensures the result is a valid probability density function, as long as every term on the right-hand side is defined as a conditional probability density function. It's typical to define $p(x_k|x_{k-1}, ..., x_1)$ as a Gaussian distribution, characterized by a mean $\mu_k = \mu_k(x_{k-1}, ..., x_1)$ and a standard deviation $\sigma_k = \sigma_k(x_{k-1}, ..., x_1)$. There's a wide range of possible choices for μ_k and σ_k, with linear models being one of the simplest classical approaches. The model is optimized by maximizing the log-likelihood

$$\log p(\boldsymbol{x}) = \sum_{k=1}^{d} \log p(x_k|x_{k-1}, ..., x_1) = d \log(2\pi) - \sum_{k=1}^{d} \left(\frac{(x_k - \mu_k)^2}{2{\sigma_k}^2} + \log(\sigma_k) \right), \tag{1.3}$$

on the sampled data $\boldsymbol{x}$. While these types of models can exhibit considerable flexibility, they are hindered by the specific structure of dependencies between feature dimensions, a structure which is imposed by the architecture of the model. Furthermore, their potential may be constrained by suboptimal selections of the conditional probability family $p(x_k|x_{k-1}, ..., x_1)$, and the parameterization models employed therein.

Non-Parametric Methods:

On the contrary, non-parametric methodologies don't necessitate the data to conform to specific parametric families of probability distributions, and they usually do not presuppose a static model structure. Instead, the model's size typically expands to cater to the data's complexity and the size of the sample. Examples of frequently employed non-parametric techniques include:

Kernel Density Estimation (KDE): KDE [Parzen, 1962] involves placing a kernel function at each data point and summing them to obtain the density estimate:

$$f(\boldsymbol{x}, h) = \frac{1}{nh^d} \sum_{k=1}^{n} K(\frac{\boldsymbol{x} - \boldsymbol{X}_i}{h}), \text{ where } \int_{\mathbb{R}^d} K(\boldsymbol{x}) d\boldsymbol{x} = 1, \tag{1.4}$$

d is the dimensionality of the data, and n is the number of data points. Nevertheless, these models tend to have somewhat restricted flexibility. This becomes apparent when the Gaussian kernel is chosen, a common selection. In such scenarios, these models coincide with a mixture of isotropic Gaussians with identical variance, where the number of components is the same of the number of training points, while the mean of each component is a training point.

Dirichlet Process Mixture Models: Infinite hierarchical Bayesian mixture models [Rasmussen, 1999] allow us to model a data distribution using a mixture model without having to predefine the number of components. In setting up this model, we have to make assumptions about the prior distribution of component parameters ($\boldsymbol{\theta}$) [Neal, 2000; Li, 2019]. After deriving the posteriors of the relevant parameters, incorporating the observed data, we can perform sampling using the Gibbs sampling Markov Chain Monte Carlo method. Following a substantial burn-in period, the parameters sampled in successive iterations are likely to display minimal variation, indicating convergence. These stabilized parameters can then serve as an estimate of the posterior mixture.
While Bayesian mixture models offer robust capabilities for density estimation and clustering, and can automatically discern the number of clusters while fitting intricate, multi-modal distributions, they do present challenges. Determining the convergence of the Markov Chain Monte Carlo (MCMC) techniques, such as Gibbs sampling, used for parameter estimation, can be non-trivial. Moreover, given the outcome is still a mixture model, the model complexity remains high in high-dimensional data, leading to intensive computational demands during the training phase [Meguelati, 2019].

Orthogonal Series: Orthogonal series methods [Silverman, 1986; Anderson, 1980] for density estimation involve approximating the density function using an orthogonal basis of functions. These basis functions can be of various types such as Fourier series, wavelets, etc. The Fourier series version is provided below:

$$f(\boldsymbol{x}) = \sum_{\substack{\boldsymbol{p} \in \{0,1,\ldots,P\}^d \\ \max_i p_i \leq P}} c_{p_1 \ldots p_d} \exp(i 2\pi \boldsymbol{p} \cdot \boldsymbol{x}), \tag{1.5}$$

where P depends on the number of samples N. Setting $c_{0,0,\ldots,0} = 1$ ensures that $f(\boldsymbol{x})$ integrates to one, however, ensuring non-negativity is not trivial. Furthermore, as it can be seen the number of parameters to estimate grows exponentially with the dimension of the data d.

1.1.2 Deep Learning (DL) Methods

Deep learning facilitates the construction of computational models that incorporate multiple processing layers. These models are recognized as universal approximators, capable of accurately simulating the behavior of virtually any function. They have notably elevated the level of precision in diverse fields such as speech and visual object detection, and generative tasks to name a few [Zhao, 2019; Bond-Taylor, 2021; Mehrish, 2023]. Deep learning models offer the potential to identify intricate structures within large datasets, once trained through backpropagation [LeCun, 2015; Goodfellow, 2016]. Despite these advantages neural network functions are not necessarily probability density functions, that is, there are no guarantees that the function they define is non-negative and integrates to one. For this reason, general frameworks have been developed, in which neural networks are used to parameterize parts of the model.

Autoregressive Normalizing Flows

In Section 1.1.1, we provided a brief explanation of density estimation via Autoregressive Models. As a special case, the distribution of each dimension x_k was modelled as a conditional Gaussian, where the conditioning is over the dimensions with smaller indices. Using reparameterization, such an x_k can be written as

$$x_k = \mu_k(x_{k-1}, ..., x_1) + z_k \sigma_k(x_{k-1}, ..., x_1) = \mu_k + z_k \sigma_k, \tag{1.6}$$

where z_k follows a standard normal distribution, [Kingma, 2016; Papamakarios, 2017]. We can simultaneously write the expression for all dimensions in vectorized form:

$$\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{z} \odot \boldsymbol{\sigma}, \tag{1.7}$$

for $\boldsymbol{\mu} = \{\mu_1, \mu_2, ..., \mu_d\}$, $\boldsymbol{\sigma} = \{\sigma_1, \sigma_2, ..., \sigma_d\}$ and $\boldsymbol{z} = \{z_1, z_2, ..., z_d\}$. As explained, the goal is to maximize the log-likelihood

$$\log p(\boldsymbol{x}) = -d \log(2\pi) - \sum_{k=1}^{d} \left(\frac{(x_k - \mu_k)^2}{2{\sigma_k}^2} + \log(\sigma_k) \right). \tag{1.8}$$

We notice that we can rewrite this equation as

$$\log p(\boldsymbol{x}) = -\left(d \log(2\pi) + \sum_{k=1}^{d} \frac{(x_k - \mu_k)^2}{2{\sigma_k}^2} \right) + \sum_{k=1}^{d} \log(\frac{1}{\sigma_k}). \tag{1.9}$$

Considering that $z_k = \frac{x_k - \mu_k}{\sigma_k}$ and its implication that $\frac{dz_k}{dx_k} = \frac{1}{\sigma_k}$, Equation (1.9) becomes

$$\log p(\boldsymbol{x}) = -\left(d \log(2\pi) + \sum_{k=1}^{d} \frac{z_k^2}{2} \right) + \sum_{k=1}^{d} \log(\frac{dz_k}{dx_k}). \tag{1.10}$$

The first term in Equation (1.10) can be written as:

$$\sum_{k=1}^{d}\left(\log(\frac{1}{2\pi})-\frac{z_k^2}{2}\right)=\log\prod_{k=1}^{d}\frac{1}{2\pi}e^{-\frac{z_k^2}{2}}=\log\prod_{k=1}^{d}p(z_k)=\log p(\boldsymbol{z}). \tag{1.11}$$

Regarding the second term, we recall that since μ_i, σ_i do not depend on x_j for $i \leq j$, then the Jacobian $\frac{d\boldsymbol{z}}{d\boldsymbol{x}} = \frac{d((\boldsymbol{x}-\boldsymbol{\mu})/\boldsymbol{\sigma})}{d\boldsymbol{x}}$ is upper triangular (the division of the vectors is dimension-wise). Thus the determinant of $\frac{d\boldsymbol{z}}{d\boldsymbol{x}}$ is the product of the diagonal elements:

$$\log\left|\det\left(\frac{d\boldsymbol{z}}{d\boldsymbol{x}}\right)\right|=\log\prod_{k=1}^{d}\frac{dz_k}{dx_k}=\sum_{k=1}^{d}\log(\frac{dz_k}{dx_k}). \tag{1.12}$$

Based on Equations (1.11) and (1.12), then Equation (1.10) becomes

$$\log p(\boldsymbol{x})=\log p(\boldsymbol{z})+\log\left|\det\left(\frac{d\boldsymbol{z}}{d\boldsymbol{x}}\right)\right|, \tag{1.13}$$

which is the logarithmic version of the change of variable formula:

$$p(\boldsymbol{x})=p(\boldsymbol{z})\left|\det\left(\frac{d\boldsymbol{z}}{d\boldsymbol{x}}\right)\right|. \tag{1.14}$$

The variables $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$ are parametrized by deep neural networks with intelligently designed architectures that respect the autoregressive formulation while maximizing flexibility, [Papamakarios, 2017; Van den Oord, 2016]. Since autoregressive models are invertible by construction and since neural networks are differentiable, then these transformations are diffeomorphisms and the use of change of variable is justified. In the deep learning setting, models that leverage the change of variable formula are collectively known as Normalizing Flows.

Normalizing Flows

The Normalizing Flow framework was previously defined in [Tabak, 2010; Tabak, 2013], and was popularised by [Rezende, 2015] and by [Dinh, 2015] respectively in the context of variational inference and density estimation.
A Normalizing Flow is a transformation defined by a sequence of invertible and differentiable functions mapping a simple base probability distribution (e.g., a standard normal) into a more complex one. Let $\boldsymbol{Z}$ and $\boldsymbol{X} = g(\boldsymbol{Z})$ be random variables where g is a diffeomorphism with inverse h. If we denote their probability density functions by $f_{\boldsymbol{Z}}$ and $f_{\boldsymbol{X}}$, based on the change of variable theorem we get:

$$f_{\boldsymbol{X}}(\boldsymbol{x})=f_{\boldsymbol{Z}}(\boldsymbol{z})\left|\det\left(\frac{d\boldsymbol{z}}{d\boldsymbol{x}}\right)\right|=f_{\boldsymbol{Z}}(h(\boldsymbol{x}))\left|\det\left(\frac{dh(\boldsymbol{x})}{d\boldsymbol{x}}\right)\right|. \tag{1.15}$$

In general, we want to optimize the parameters of h such that we maximize the likelihood of sampled points $\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(m)}$. Once these parameters are optimized, then we can give

as input any test point $\boldsymbol{x}$ on the right hand side of Equation (1.15), and calculate its likelihood. For the generative task, being able to easily recover g from h is essential, as the generated point $\boldsymbol{x}_g$ will take form $\boldsymbol{x}_g{=}g(\boldsymbol{z}_s)$, where $\boldsymbol{z}_s$ is a sampled point from the base distribution $f_{\boldsymbol{Z}}$.
For increased modeling flexibility, we can use a chain (flow) of transformations, $\boldsymbol{x}_{i-1} = g_i(\boldsymbol{x}_i),\ i \in [n]$, that is $\boldsymbol{x}_i = h_i(\boldsymbol{x}_{i-1}),\ i \in [n]$. In this case, due to chain rule we have:

$$f_{\boldsymbol{X}_0}(\boldsymbol{x}_0) = f_{\boldsymbol{X}_n}(\boldsymbol{x}_n)\left|\det\left(\frac{d\boldsymbol{x}_n}{d\boldsymbol{x}_0}\right)\right| = f_{\boldsymbol{X}_n}(h_n(...h_1(\boldsymbol{x}_0)))\prod_{i=1}^{n}\left|\det\left(\frac{dh_i(\boldsymbol{x}_{i-1})}{d\boldsymbol{x}_{i-1}}\right)\right|, \quad (1.16)$$

where $\boldsymbol{x}_0$ is a data point. The interested reader can find a more in-depth review of normalizing flows in [Kobyzev, 2021] and [Papamakarios, 2021].

Continuous Normalizing Flows (CNFs)

While the practical application of normalizing flows is generally challenging due to computational bottlenecks, most notably regarding the $\mathcal{O}(D^3)$ computation cost of the Jacobian determinant, different architectures have been proposed in order to scale normalizing flows to high dimensions while at the same time ensuring the flexibility and bijectivity of the transformations [Rezende, 2015; Dinh, 2017; Kobyzev, 2021]. The common strategy consists of placing different architectural restrictions on the model (e.g. autoregressive case), to enforce special Jacobian forms, with less computationally demanding determinants. Neural ODEs, [Chen, 2018], are continuous generalizations of residual networks:

$$\boldsymbol{x}_{t_{i+1}} = \boldsymbol{x}_i + \epsilon f(\boldsymbol{x}_{t_i}, t_i, \boldsymbol{\theta}) \rightarrow \boldsymbol{x}(t) = \boldsymbol{x}(0) + \int_0^t f(\boldsymbol{x}(\tau), \tau, \boldsymbol{\theta})d\tau, \text{ as } \epsilon \rightarrow 0. \quad (1.17)$$

In this case, the number of compositions of transformations in Equation (1.17) goes to infinity, and the change of variable formula evolves into the instantaneous change of variable [Chen, 2018], which enables one to train continuous normalizing flows:

$$\log p(\boldsymbol{x}(0)) = \log p(\boldsymbol{x}(T)) + \int_0^T tr\frac{\partial f(\boldsymbol{x}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{x}(t)}dt, \quad (1.18)$$

where $\boldsymbol{x}(0)$ represents a sample from the data. A surprising benefit is that one does not need to calculate the determinant of the Jacobian of the transformation anymore, but simply the trace of a matrix without placing constraints in the form of the Jacobian of the overall transformation. These models are collectively called Neural ODE flows (NODEFs) or simply Continuous Normalizing Flows (CNFs). In [Grathwohl, 2019], these ideas are further explored and computational simplifications are introduced, notably the use of Hutchinson's trace estimator, [Hutchinson, 1990; Adams, 2018], as an unbiased stochastic estimator of the trace in the likelihood expression in Equation (1.18), that reduces the computation cost of the Jacobian determinant to $\mathcal{O}(D)$. The resulting model is named FFJORD.

Diffusion Probabilistic Models (DPMs)

As described, CNFs are flexible normalizing flows with free-form continuous dynamics. However, despite the Jacobian determinant's computational complexity being lowered to $\mathcal{O}(D)$ in Equation (1.18) and the memory conservation achieved through the adjoint method [Chen, 2018], the optimization task in CNFs remains challenging. These flows are trained via maximum likelihood, which can be suboptimal due to the need for the neural network to craft a transformation mapping the data distribution to a standard normal one, and due to the increase of the numerical integration steps during training. Diffusion Probabilistic Models address these concerns. A forward diffusion process or diffusion process [Sohl-Dickstein, 2015] is a fixed Markov chain that gradually adds Gaussian noise to the data according to a schedule $\{b_t | t \in [n]\}$:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) := \mathcal{N}(\boldsymbol{x}_{t-1}\sqrt{1-b_t}, b_t\boldsymbol{I}). \tag{1.19}$$

Such a process transforms the data distribution into a standard multivariate normal distribution. If we denote $a_t = 1 - b_t$ and $\bar{a}_t = \prod_{s=1}^{t} a_s$, then we can write:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0\sqrt{\bar{a}_t}, (1-\bar{a}_t)\boldsymbol{I}). \tag{1.20}$$

The reverse process conditioned on the initial sample is also described by a chain of Gaussian distributions:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}(\mu(\boldsymbol{x}_t, \boldsymbol{x}_0), \boldsymbol{\Sigma}(t)), \tag{1.21}$$

where

$$\mu(\boldsymbol{x}_t, \boldsymbol{x}_0) = \mu(\boldsymbol{x}_t, \boldsymbol{x}_0(\boldsymbol{x}_t, \boldsymbol{\varepsilon})) = \frac{1}{\sqrt{a_t}}(\boldsymbol{x}_t - \frac{b_t}{\sqrt{1-\bar{a}_t}}\boldsymbol{\varepsilon}),$$
$$\boldsymbol{\Sigma}(t) = b_t\boldsymbol{I}.$$

The goal is to approximate this reverse process via

$$p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}(\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}(t)) \tag{1.22}$$

that converts the standard multivariate normal distribution into the data distribution.

To this end, for each step t, the KL divergence between $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ and $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is minimized, which amounts to minimizing

$$\mathbb{E}_{\boldsymbol{x_0}, \boldsymbol{x_t}}||\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \mu(\boldsymbol{x}_0, \boldsymbol{x}_t)||^2, \tag{1.23}$$

or equivalently

$$\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{\varepsilon}}||\boldsymbol{\varepsilon}_\theta(\boldsymbol{x_0}\sqrt{\bar{a}_t} + \sqrt{(1-\bar{a}_t)}\boldsymbol{\varepsilon}, t) - \boldsymbol{\varepsilon}||^2, \tag{1.24}$$

for $\varepsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, data samples $\boldsymbol{x_0}$, and a neural network $\varepsilon_\theta(\boldsymbol{x_t}, t)$ [Ho, 2020].

SDE Diffusion Models and their CNF Representation

For an $\varepsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, the forward diffusion process defined in Equation (1.19) can be written as

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1}\sqrt{1-b_t} + \sqrt{b_t}\boldsymbol{\varepsilon}. \tag{1.25}$$

As derived in [Song, 2021], the continuous counterpart of this process takes the form

$$d\boldsymbol{x}(t) = -\frac{1}{2}b(t)\boldsymbol{x}(t)dt + \sqrt{b(t)}d\boldsymbol{w}. \tag{1.26}$$

In this case Equation (1.20) becomes

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0\mu_t, \sigma_t^2\boldsymbol{I}), \tag{1.27}$$

where $\mu_t = e^{-\frac{1}{2}\int_0^t b(s)ds}$ and $\sigma_t = \sqrt{(1 - e^{-\int_0^t b(s)ds})}$. As shown through the Fokker-Plack equation, the evolution of the probability density function of the data, as dictated by the FDP, is identical to the evolution dictated by the following ODE transformation:

$$d\boldsymbol{x}(t) = -\frac{1}{2}b(t)[\boldsymbol{x}(t) + \nabla_{\boldsymbol{x}(t)} \log p_t(\boldsymbol{x}(t))]dt = f_t(\boldsymbol{x}(t))dt. \tag{1.28}$$

Knowing f_t allows us to perform data generation and likelihood estimation, through the framework of continuous normalizing flows. It can be observed that the only unknown in f_t, is the score $\nabla_{\boldsymbol{x}(t)} \log p_t(\boldsymbol{x}(t))$. This quantity can be modelled using a neural network $\boldsymbol{s}_\theta(\boldsymbol{x}(t), t)$ trained either through sliced score matching [Song, 2020]:

$$\min\ \mathbb{E}_{\boldsymbol{x}_t \sim p(\boldsymbol{x}_t)}[\frac{1}{2}||\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)||^2 + div(\boldsymbol{s}_\theta(\boldsymbol{x}_t, t))] \tag{1.29}$$

or the equivalent MSE denoising loss [Ho, 2020; Kingma, 2021]:

$$\min\ \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{\varepsilon}}||\boldsymbol{s}_\theta(\boldsymbol{x}_0\mu_t + \sigma_t\boldsymbol{\varepsilon}, t) - (-\frac{\boldsymbol{\varepsilon}}{\sigma_t})||^2. \tag{1.30}$$

We notice that the loss function is now an MSE and that the network does not have to devise a process that turns the data distribution into a standard normal one, as such a process is generated by the forward diffusion process. These properties of DPMs simplify the learning task significantly.

1.2 Tail Shape Estimation of Probability Distributions

Tails are an integral part of probability density functions, indicating infrequent but impactful outcomes. Their thickness estimates the probability of extreme events and pro-

vides information about the moments of the distribution. Thin tails, as seen in a normal distribution, imply lower probabilities of extreme events, much like the distribution of heights in people. In contrast, heavy tails suggest higher possibilities of outliers.

While density estimation methods such as Normalizing Flows and Diffusion Models, as described in Section 1.1, work well when modeling areas of high probability that are adequately represented by a large number of training data points, these models do not perform as effectively in modeling low probability areas [Song, 2019], such as the tails of distributions.

Because extreme events, by definition, occur infrequently, extrapolating based on averages of typical events can lead to an underestimation of their likelihood. Extreme Value Theory (EVT) provides a rigorous statistical framework for modeling the tails of distributions to more accurately estimate rare extreme occurrences.

1.2.1 Extreme Value Theory

Extreme Value Theory (EVT) is an established field concerned with modelling the tails of distributions. It dates back to 1923, when Richard von Mises discovered that the Gumbell distribution is the limiting distribution of the maximum of an iid sequence, sampled from a Gaussian distribution. In 1928, Ronald A. Fisher and Leonard H. C. Tippett in [Fisher, 1928], characterized the only three possible non-degenerate limiting distributions of the maximum in the general case: Frechet, Gumbel and Weibull. In 1943, Boris V. Gnedenko, gave a rigorous proof of this fact in [Gnedenko, 1943]. This result is known Fisher–Tippett–Gnedenko theorem, and forms the foundation of EVT. The three aforementioned limiting distributions of the maximum can be written in compact form and they are known as the class of extreme value distributions:

Definition 1.2.1. *The Generalized Extreme Value Distribution (GEVD) is defined as follows:*

$$G_{\xi,a,b}(x) = e^{-(1+\xi(ax+b))^{-\frac{1}{\xi}}}, \quad 1+\xi(ax+b) > 0, \tag{1.31}$$

where $b \in \mathbb{R}$, $\xi \in \mathbb{R} \setminus \{0\}$ and $a > 0$. For $\xi = 0$, we define the generalized Extreme Value Distribution as the limit when $\xi \to 0$, that is

$$G_{0,a,b}(x) = e^{-e^{-ax-b}}. \tag{1.32}$$

If the limiting distribution of the maximal sample from a distribution $F(x)$ is $G_{\xi,a,b}(x)$ as defined in Definition 1.2.1, then depending on whether $\xi > 0$, $\xi = 0$, $\xi < 0$, we say that F is in the MDA (Maximum Domain of Attraction) of a Frechet, Gumbell, or Weibull

distribution respectively. The measure of the thickness of a tail is ξ, that is, a distribution $F(x)$ has a thick tail if its corresponding ξ is positive, thin tail if $\xi = 0$, and no tail if $\xi < 0$. This means that distributions, whose maximal sample converges in distribution to a Frechet limiting distribution, are characterized by thick tails, indicating a higher probability of extreme events. On the other hand, distributions with a limiting Gumbel distribution have thin tails, suggesting a lower likelihood of extreme values. Finally, the Weibull distribution is the limiting distribution of the maximal sample from a distribution with bounded support.

Considering the discussion above, it is important to be able to estimate properly the shape parameter ξ given a set of samples from a distribution. One of the most direct methods makes use of the Fisher–Tippett–Gnedenko theorem, by dividing the samples into non-overlapping blocks of equal size and then by picking the maximum observation in each block. The new observations created, assuming a significant number of samples are contained in each block, follow approximately an extreme value distribution $G_{\xi,a,b}(x)$ for some real ξ. Parametric statistical methods for the extreme value distributions are then applied to those observations. This approach is known as the *block maxima* method [Gumbel, 1958]. There are two major sets of estimators that are widely used in this method [Ferreira, 2015]: the maximum likelihood estimators [Prescott, 1980] and the probability weighted moment (PWM) estimators [Hosking, 1985].

The block maxima approach offers benefits when observations show dependence within blocks yet independence between blocks (e.g., seasonal periodicity for annual maxima). As periodic blocks naturally arise in many situations, this method is often fitting. However, inherent drawbacks exist, as this method omits some high observations while retaining lower values. By retaining only the maximum per block, the approach misses certain peak data points while including suboptimal (low-valued) data [Ferreira, 2015].

The Peak-Over-Threshold (POT) method, the approach we will focus on in this thesis, is another important approach for estimating the shape parameter of tails. The foundation of the POT method is the Pickands–Balkema–De Haan theorem [Pickands, 1975; Balkema, 1974]. This theorem roughly states that for a distribution in the max-domain of attraction of a generalized extreme value distribution, the limiting distribution of samples exceeding a threshold converges to a generalized Pareto distribution with a location parameter of zero. Furthermore, the theorem proves that the shape parameters of the generalized extreme value distribution and corresponding generalized Pareto distribution are equal. In simpler terms, the theorem states that for a large class of distributions, samples from a distribution of this class that surpass a high enough threshold follow a generalized Pareto distribution. The shape parameter of the generalized pareto can then be estimated using methods such as the Pickands estimator or the MLE based Deckers-Einmahl-de Haan estimator [Dekkers, 1989a] on the samples above the threshold. Whereas the block maxima approach may exclude higher observations from one block, and potentially favour

a low valued maximum of another block, the POT method retains solely exceedances over a set threshold.

The previously described methods were designed for application to general distributions. However, the distribution of interest may sometimes be expressed as a marginal distribution derived from dependent random variables. A relevant example comes from the machine learning domain, where we may be interested in studying the distribution of statistics of interest (e.g. the mean squared loss) in a cross-validation setting. In this case, the estimation of each sample is conditioned on the training/testing split, and we would like to characterise the shape of the loss distribution under this conditional dependency regime. In this case, classical tail estimation methods may suffer of important limitations, and novel approaches incorporating the additional information from such dependent variables should be devised to enhance tail shape estimation for the variable of interest. This challenge is developed in the following subsection.

1.2.2 Tails of Marginal Distributions

In a general machine learning problem, we assume that each data sample $(\boldsymbol{X}, \boldsymbol{Y})$ comes from distribution $\mathcal{D}$ and that the sampling is independent. We have used the symbol $\boldsymbol{X}$ to denote the features and the symbol $\boldsymbol{Y}$ to denote the labels. The training set will be defined as a random vector comprised of iid random vectors $(\boldsymbol{X}, \boldsymbol{Y})$ sampled from $\mathcal{D}$. More precisely, after fixing a natural number k, we define a training set as $\boldsymbol{V} = [(\boldsymbol{X}, \boldsymbol{Y})_1, (\boldsymbol{X}, \boldsymbol{Y})_2, ..., (\boldsymbol{X}, \boldsymbol{Y})_k]$, where each $(\boldsymbol{X}, \boldsymbol{Y})_i$ has distribution $\mathcal{D}$. On the other hand, a test point $\boldsymbol{U}$ naturally is defined as a sample from $\mathcal{D}$, i.e., $\boldsymbol{U} = (\boldsymbol{X}, \boldsymbol{Y})$. In practice, the realisation of $\boldsymbol{U}$ should not be an entry in $\boldsymbol{V}$.
A model which is trained on $\boldsymbol{V}$ to predict $\boldsymbol{Y}$ from $\boldsymbol{X}$ is denoted as $\hat{\boldsymbol{h}}_{\boldsymbol{V}}(\boldsymbol{X})$. The prediction error on the testing datum $\boldsymbol{U}$ of a model trained on $\boldsymbol{V}$ is denoted as $W_{\boldsymbol{V}}(\boldsymbol{U})$. We notice that the probability density function of $W_{\boldsymbol{V}}(\boldsymbol{U})$ is

$$f_W(w) = \int f_{W,\boldsymbol{V}}(w, \boldsymbol{v})d\boldsymbol{v} = \int f_{\boldsymbol{V}}(\boldsymbol{v}) f(w|\boldsymbol{V} = \boldsymbol{v})d\boldsymbol{v} = \int f_{\boldsymbol{V}}(\boldsymbol{v}) f_{\boldsymbol{v}}(w)d\boldsymbol{v}, \tag{1.33}$$

therefore the distribution function of $W_{\boldsymbol{V}}(\boldsymbol{U})$ is:

$$F_W(w) = \int f_{\boldsymbol{V}}(\boldsymbol{v}) F_{\boldsymbol{v}}(w)d\boldsymbol{v}. \tag{1.34}$$

$F_{\boldsymbol{v}}(w)$ is the distribution of the prediction error (loss) of the model trained on training set $\boldsymbol{v}$, while $F_W(w)$ is the unconditional distribution of the loss. A natural question is whether the tail shape parameters of the conditional distributions $F_{\boldsymbol{v}}$ contain information about the shape tail parameter of F_W.

This question is important, as potential issues can occur when applying POT directly on samples of the marginal distribution rather than using the conditional dependence structure. The main reason why direct application of POT can fail to properly estimate the tail is that, since the marginal by definition is the mixture of many (potentially infinite) distributions, the non-tail part of some distributions can overlap and overshadow the tails of others. This is in particular the case when the medians of components with thin tails are located in the tail part of thick-tailed components. As we are more likely to sample from the high-likelihood regions of the thin-tailed components where the medians are located, rather than from the thick tails of other components, our sample set could fail to appropriately describe the true shape of the tail of the marginal. In Section 1.3.2, we provide a more detailed explanation of this challenge, which motivates one of the contributions of this thesis.

1.3 Objectives and Organization of the Thesis

In this thesis, we aim to resolve some important challenges that arise in the field of density estimation and tail shape estimation of marginal distributions. In Section 1.3.1 we lay out two such challenges in the task of density estimation, which we tackle in Chapter 2 and 3 respectively, which in turn comprise **Part I** of the thesis. The third challenge regarding tail-shape estimation is presented in Section 1.3.2, and is resolved in Chapter 4 presented in **Part II** of this thesis.

Finally, we conclude the manuscript in Chapter 5 by summarizing the main contributions of this work and exploring further research possibilities.

1.3.1 Density Estimation Challenges via Modern DL Methods

Continuous Normalizing Flows and Diffusion Probabilistic Models present formidable approaches for modelling high-dimensional data distributions and achieving accurate density estimations. However, they are not without certain constraints. Firstly, these models necessitate the integration of both the point $\boldsymbol{x}_t$ and the time variable t as inputs for the network operating at time t. As such, it is of critical importance to strategically implement the time component, thereby enabling a flexible network evolution over time. Moreover, while diffusion models considerably streamline the training process and mitigate memory requirements by foregoing the need for maximum likelihood loss, it is vital to emphasize that standard training methodologies do not optimally leverage the inherent characteristics of diffusion models to boost flexibility and curtail training duration. A key underutilized feature is the independence of the optimization tasks at distinct time points.

Challenge I: Temporal Flexibility of the Vector Field in the CNF Context

While the neural ODE framework of normalizing flows, such as the one used in FFJORD, allows us to compute determinants of free form Jacobians in $\mathcal{O}(D)$ time, the flexibility of the transformations engendered by neural ODEs has proven to be less than ideal. Within the CNF paradigm, both the point $\boldsymbol{x}_t$ and the time variable t are requisite inputs for the network $f_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ operating at time t. As a result, it is of paramount importance to implement the time component in a strategic manner, thus ensuring temporal flexibility in network evolution.

To address this constraint, in Chapter 2, we introduce AFFJORD, a neural ODE-based normalizing flow, which bolsters the representational capability of FFJORD by defining the neural ODE through specially augmented transformation dynamics that conserve the topological properties of the space. Our experimental evaluations on density estimation in synthetic and high-dimensional data such as MNIST, CIFAR-10, and CelebA (32×32) evidence that AFFJORD surpasses the baseline FFJORD by virtue of enhanced flexibility in the underlying vector field. Furthermore, we derive the Jacobian determinant of the general augmented form by extending the continuous chain rule into the *cable rule*, which formulates the forward sensitivity of ODEs in relation to their initial conditions. This cable rule provides an explicit formula for the Jacobian of a neural ODE transformation and offers an elegant proof for the instantaneous change of variable.

Challenge II: Modeling Vector Field Evolution through a Single Network

Contrary to the CNF framework, in Diffusion Probabilistic Models (DPM), the network that models the vector field employs the MSE loss as described in Equation (1.30), which considerably accelerates the training. Furthermore, diffusion models alleviate the intricacy of learning the evolutionary process, as the forward diffusion process outlines the temporal evolution of the data distribution, thus reducing the framework's role to merely modeling the predefined vector fields. Nevertheless, if parameterized by a single time-varying network, these models still encounter limited flexibility, as a solitary set of parameters (or model) is charged with modeling all vector fields defined by the Forward Diffusion Process (FDP). Moreover, the optimization of a vector field at time t is contingent upon the optimization of other vector fields at different time points, thereby impeding the speed of the training process.

To counter these issues, in Chapter 3, we suggest harnessing the independence of learning tasks at different time points inherent to DPMs. More precisely, we segment the learning task by employing independent networks, each specifically designed to learn the evolution of scores within a particular time sub-interval. Additionally, taking a cue from residual flows, we extend this strategy by using separate networks to independently model the score at each discrete time point. As substantiated by empirical evidence on synthetic

and image datasets, our methodology not only significantly speeds-up the training process by introducing an extra layer of parallelization on top of data parallelization, but it also improves density estimation performance relative to the conventional training methodology for DPMs

1.3.2 Tail-shape Estimation Challenges of Marginal Distributions

The direct application of the POT method on $F_W(w)$ (Equation (1.34)) can present challenges due to the variability of the locations of the tails of conditional distributions that underlie the marginal. In what follows, we provide an intuitive explanation as to why this is the case, and also summarize our approach to alleviating this problem.

Challenge III: Variability in tail locations of conditional distributions underlying the marginal

To gain some insight into the problem, we study the following simple example. We assume that our variable of interest is $X > 0$, which in turns depends on the variable Z. Our goal is to estimate the tail shape parameter of the distribution of X, that is $F_X(x)$. For simplicity we can assume that Z can be either 0 or 1, with equal probability, and if $Z = 0$ then $f_0(x) = f(x|Z = 0)$ is a thick tailed distribution (shape parameter $\xi_0 > 0$), while if $Z = 1$ then $f_1(x) = f(x|Z = 1)$ is a thin tailed distribution ($\xi_1 = 0$), with a large but finite mean. The (marginal) distribution of X is

$$f(x) = \frac{1}{2}f_0(x) + \frac{1}{2}f_1(x), \text{ that is } F(x) = \frac{1}{2}F_0(x) + \frac{1}{2}F_1(x). \tag{1.35}$$

Given a sample set $\{(x^{(1)}, z^{(1)}), (x^{(2)}, z^{(2)}), ..., (x^{(m)}, z^{(m)})\}$, one approach is to simply ignore the second dimension of each point in the sample set, and directly use the POT method on the points $\{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$, in order to estimate the tail shape parameter of $f(x)$. However, for small m, since the thin tailed component in the mixture distribution in Equation (1.35) has a very large mean, we expect most of the large values in $\{x^{(1)}, x^{(2)}, ..., x^{(m)}\}$, to come from this component. This is due to the fact that we are unlikely to sample many values $x^{(i)}$ from the thick tail of $f_0(x)$ exceeding the mean of $f_1(x)$. This implies that the sample tail of $f(x)$ will be defined by the samples from the thin tailed $f_1(x)$, while the true tail shape ξ of $f(x)$ is known to be precisely the tail shape parameter of the distribution with the thickest tail, that is, ξ_0. This motivates the idea of not discarding the dependent values $z^{(i)}$ by marginalizing over Z, but to incorporate such information in the tail shape estimation process.

The discussed problem can be generalized when the variable $\boldsymbol{Z}$ is continuous, that is,

$$F_X(x) = \int f_{\boldsymbol{Z}}(\boldsymbol{z}) F_{\boldsymbol{z}}(x) d\boldsymbol{z}, \tag{1.36}$$

where we see that this formulation covers the setting we encountered in Section 1.2.2 (Equation (1.34)).

In Chapter 4, we develop a general method to mitigate the issue of estimating the tails of marginal distributions, when there exists a large variability between locations of the individual conditional distributions underlying the marginal. The proposed solution enables a reduction in the sample size requirements, in the experiments we conducted. To this end, we demonstrate that under some regularity conditions, the shape parameter of the marginal distribution is precisely the maximum tail shape parameter of the family of conditional distributions. We refer to the method constructed from this result as *cross tail estimation*, due to similarities that it shares with Monte Carlo cross validation. An additional benefit of using the approach proposed here instead of the standard POT, is the reduced computational time in the case that the marginal is estimated from many conditional distributions. In the context of the distributions of loss functions (Equation (1.34)), our theory establishes that, under some assumptions, we can estimate the shape of the total loss distribution, by simply investigating the models prediction, without the need for target data. Furthermore, we show evidence of polynomial decay of tails of distributions of model predictions, and empirically demonstrate a relationship between the thickness of such tails and overfitting.

1.4 Publications

The described contributions led to the following peer-reviewed publications:

- Etrit Haxholli, Marco Lorenzi. On Tail Decay Rate Estimation of Loss Function Distributions. *Journal of Machine Learning Research*, In press, (2023).

- Etrit Haxholli, Marco Lorenzi. Enhanced Distribution Modelling via Augmented Architectures For Neural ODE Flows. *DLDE-III Workshop in the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*, Dec 2023, New Orleans, Louisiana, United States.

- Etrit Haxholli, Marco Lorenzi. Faster Training and Improved Performance of Diffusion Models via Parallel Score Matching. *NeurIPS 2023 Workshop on Diffusion Models*, Dec 2023, New Orleans, Louisiana, United States.

Part I

Density Estimation

2 Enhanced Distribution Modelling via Augmented Architectures For Neural ODE Flows

Contents

In the introductory discourse, we touched upon the fact that while the neural ODE formulation of normalizing flows such as in FFJORD enables us to calculate the determinants of free form Jacobians in $\mathcal{O}(D)$ time, the flexibility of the transformation underlying neural ODEs has been shown to be suboptimal. In this chapter, we present AFFJORD, a neural ODE-based normalizing flow which enhances the representation power of FFJORD by defining the neural ODE through special augmented transformation dynamics which preserve the topology of the space. Furthermore, we derive the Jacobian determinant of the general augmented form by generalizing the chain rule in the continuous sense into the *cable rule*, which expresses the forward sensitivity of ODEs with respect to their initial conditions. The cable rule gives an explicit expression for the Jacobian of a neural ODE transformation, and provides an elegant proof of the instantaneous change of variable. Our experimental results on density estimation in synthetic and high dimensional data, such as MNIST, CIFAR-10 and CelebA (32×32), show that AFFJORD outperforms the baseline FFJORD through the improved flexibility of the underlying vector field.

2.1 Introduction

Normalizing flows are diffeomorphic random variable transformations providing a powerful theoretical framework for generative modeling and probability density estimation [Rezende, 2015]. While the practical application of normalizing flows is generally challenging due to computational bottlenecks, most notably regarding the $\mathcal{O}(D^3)$ computation cost of the Jacobian determinant, different architectures have been proposed in order to scale normalizing flows to high dimensions while at the same time ensuring the flexibility and bijectivity of the transformations [Rezende, 2015; Dinh, 2017; Kobyzev, 2021]. The common strategy consists of placing different architectural restrictions on the model, to enforce special Jacobian forms, with less computationally demanding determinants.

A noteworthy approach is based on neural ODEs [Chen, 2018], as they enable us to calculate the determinants of free form Jacobians in $\mathcal{O}(D)$ time [Grathwohl, 2019]. More specifically, the rule for the instantaneous change of variable [Chen, 2018] provides an important theoretical contribution to normalizing flows, as it yields a closed form expression of the Jacobian determinant of a neural ODE transformation. In this case, calculating the Jacobian determinant simplifies to calculating the integral of the divergence of the vector field along the transformation trajectory. Such models are known as Continuous Normalizing Flows (CNFs). In [Grathwohl, 2019], these ideas are further explored and computational simplifications are introduced, notably the use of Hutchinson's trace estimator [Hutchinson, 1990]. The resulting model is named FFJORD.

In [Dupont, 2019], it is shown that there exist functions which neural ODEs are not capable of representing. To tackle this issue and enhance the expressiveness of the neural ODE transformation, they propose to lift the data into a higher dimensional space, on which the neural ODE is applied, to subsequently project the output back to the original space. Such augmented neural ODEs (ANODEs) have been experimentally shown to lead to improved flexibility and generalisation properties than the non-augmented counterpart.

Inspired by [Dupont, 2019], in this chapter we develop a theoretical framework to increase the flexibility of ODE flows, such as FFJORD, through dimension augmentation. To this end, we derive an explicit formula for Jacobians corresponding to neural ODE transformations. This formula represents the continuous generalization of the chain rule, which we name *the cable rule*, that ultimately allows the derivation of the Jacobian expression and determinant for the composition of the operations defining ANODE flows: augmentation, neural ODE transformation, and projection.

To enable computational feasibility and ensure that the ANODE transformation is diffeomorphic, we allow the augmented dimensions to parameterize the vector field acting on

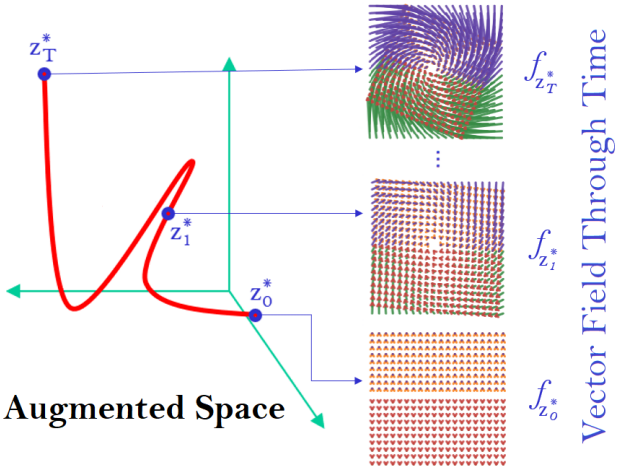

Fig. 2.1.: Vector field evolution in AFFJORD as a function of the augmented dimensions $\boldsymbol{z}_t^*$.

the original dimensions (but not vice-versa). We name the resulting model augmented FFJORD (AFFJORD). In AFFJORD, the evolution of time is high dimensional (Figure 2.1), and is learnt via the vector field defined by the augmented component, contrasting the linear time evolution of FFJORD. This setup coincides with the framework introduced by [Zhang, 2019], but in the context of normalizing flows. Our experiments on 2D data of toy distributions and image datasets such as MNIST, CIFAR-10 and CelebA (32×32), show that the proposed ANODE flow defined by AFFJORD outperforms FFJORD in terms of density estimation, thus highlighting the improved flexibility and representation properties of the underlying vector field.

2.2 Background and Related Work

Normalizing Flows: The Normalizing Flow framework was previously defined in [Tabak, 2010; Tabak, 2013], and was popularised by [Rezende, 2015] and by [Dinh, 2015] respectively in the context of variational inference and density estimation.
A Normalizing Flow is a transformation defined by a sequence of invertible and differentiable functions mapping a simple base probability distribution (e.g., a standard normal) into a more complex one. Let $\boldsymbol{Z}$ and $\boldsymbol{X} = g(\boldsymbol{Z})$ be random variables where g is a diffeomorphism with inverse h. If we denote their probability density functions by $f_{\boldsymbol{Z}}$ and $f_{\boldsymbol{X}}$, based on the change of variable theorem we get:

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = f_{\boldsymbol{Z}}(\boldsymbol{z}) \left| \det \left(\frac{d\boldsymbol{z}}{d\boldsymbol{x}} \right) \right| = f_{\boldsymbol{Z}}(h(\boldsymbol{x})) \left| \det \left(\frac{dh(\boldsymbol{x})}{d\boldsymbol{x}} \right) \right|. \tag{2.1}$$

In general, we want to optimize the parameters of h such that we maximize the likelihood of sampled points $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$. Once these parameters are optimized, then we can give as input any test point $\boldsymbol{x}$ on the right hand side of Equation (2.1), and calculate its likelihood. For the generative task, being able to easily recover g from h is essential, as

the generated point $\boldsymbol{x}_g$ will take form $\boldsymbol{x}_g$=$g(\boldsymbol{z}_s)$, where $\boldsymbol{z}_s$ is a sampled point from the base distribution $f_{\boldsymbol{Z}}$.
For increased modeling flexibility, we can use a chain (flow) of transformations, $\boldsymbol{z}_{i-1} = g_i(\boldsymbol{z}_i),\ i \in [n]$, that is $\boldsymbol{z}_i = h_i(\boldsymbol{z}_{i-1}),\ i \in [n]$. In this case due to chain rule we have:

$$f_{\boldsymbol{Z}_0}(\boldsymbol{z}_0) = f_{\boldsymbol{Z}_n}(\boldsymbol{z}_n)\left|\det\left(\frac{d\boldsymbol{z}_n}{d\boldsymbol{z}_0}\right)\right| = f_{\boldsymbol{Z}_n}(h_n(...h_1(\boldsymbol{z}_0)))\prod_{i=1}^{n}\left|\det\left(\frac{dh_i(\boldsymbol{z}_{i-1})}{d\boldsymbol{z}_{i-1}}\right)\right|, \tag{2.2}$$

where $\boldsymbol{z}_0$ is a data point. The interested reader can find a more in-depth review of normalizing flows in [Kobyzev, 2021] and [Papamakarios, 2021].

Neural ODE Flows: Neural ODEs, [Chen, 2018], are continuous generalizations of residual networks:

$$\boldsymbol{z}_{t_{i+1}} = \boldsymbol{z}_{t_i} + \epsilon f(\boldsymbol{z}_{t_i}, t_i, \boldsymbol{\theta}) \rightarrow \boldsymbol{z}(t) = \boldsymbol{z}(0) + \int_0^t f(\boldsymbol{z}(\tau), \tau, \boldsymbol{\theta})d\tau, \text{ as } \epsilon \rightarrow 0. \tag{2.3}$$

In [Pontrjagin, 1962; Chen, 2018], it is shown that the gradients of neural ODEs can be computed via the adjoint method, with constant memory cost with regards to "depth":

$$\frac{dL}{d\boldsymbol{\theta}} = \int_0^T \frac{\partial L}{\partial \boldsymbol{z}(t)}\frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}dt = -\int_T^0 \frac{\partial L}{\partial \boldsymbol{z}(t)}\frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}dt, \tag{2.4}$$

where $\frac{\partial L}{\partial \boldsymbol{z}(t)}$ can be calculated simultaneously by

$$\frac{\partial L}{\partial \boldsymbol{z}(t)} = \frac{\partial L}{\partial \boldsymbol{z}(T)} - \int_T^t \frac{\partial L}{\partial \boldsymbol{z}(\tau)}\frac{\partial f(\boldsymbol{z}(\tau), \tau, \boldsymbol{\theta}(\tau))}{\partial \boldsymbol{z}(\tau)}d\tau. \tag{2.5}$$

In addition, they also derive the expression for the instantaneous change of variable, which enables one to train continuous normalizing flows:

$$\log p(\boldsymbol{z}(0)) = \log p(\boldsymbol{z}(T)) + \int_0^T tr\frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}dt, \tag{2.6}$$

where $\boldsymbol{z}(0)$ represents a sample from the data. A surprising benefit is that one does not need to calculate the determinant of the Jacobian of the transformation anymore, but simply the trace of a matrix. These models are collectively called Neural ODE flows (NODEFs) or simply Continuous Normalizing Flows (CNFs). In [Grathwohl, 2019], these ideas are further explored and computational simplifications are introduced, notably the use of Hutchinson's trace estimator, [Hutchinson, 1990; Adams, 2018], as an unbiased stochastic estimator of the trace in the likelihood expression in Equation (2.6). The resulting model is named FFJORD.

Multiscale Architectures: In [Dinh, 2017], a multiscale architecture for normalizing flows is implemented, which transforms the data shape from $[c, s, s]$ to $[4c, \frac{s}{2}, \frac{s}{2}]$, where c is the number of channels and s is the height and width of the image. Effectively

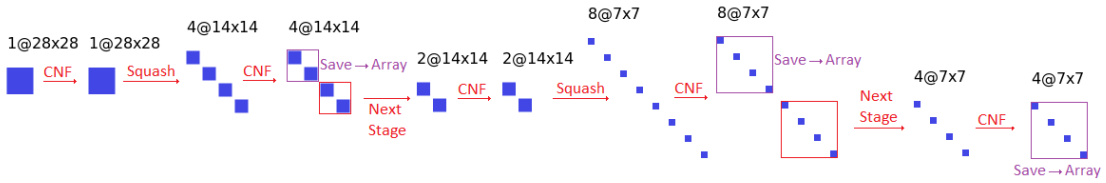

Fig. 2.2.: Example of the multiscale architecture on MNIST dataset.

this operation trades spatial size for additional channels, and after each transformation a normalizing flow is applied on half the channels, while the other half are saved in an array. This process can be repeated as many times as the width and height of the transformed image remain even numbers. In the end, all saved channels are concatenated to construct an image with the original dimensions. A visual description of this process can be found in Figure 2.2.

Augmented Neural ODEs: Considering that transformations by neural ODEs are diffeomorphisms (hence homeomorphisms), [Dupont, 2019] show that Neural Ordinary Differential Equations (NODEs) learn representations that preserve the topology of the input space, and prove that this implies the existence of functions that Neural ODEs cannot represent. To address these limitations, they introduce Augmented Neural ODEs:

$$\frac{d}{dt}\begin{bmatrix}\boldsymbol{z}(t)\\ \boldsymbol{z}^*(t)\end{bmatrix} = h\left(\begin{bmatrix}\boldsymbol{z}(t)\\ \boldsymbol{z}^*(t)\end{bmatrix}, t, \boldsymbol{\theta}\right) = \begin{bmatrix}f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})\\ g(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})\end{bmatrix}$$

$$\text{for} \quad \begin{bmatrix}\boldsymbol{z}(0)\\ \boldsymbol{z}^*(0)\end{bmatrix} = \begin{bmatrix}\boldsymbol{x}\\ 0\end{bmatrix}, \tag{2.7}$$

where $\boldsymbol{z}^*(t)$ is the augmented component, and $h = [f, g]$ is the vector field to be learnt. In addition to being more expressive models, [Dupont, 2019] show that augmented neural ODEs are empirically more stable, generalize better and have a lower computational cost than Neural ODEs. A schematic of their architecture in the discrete case can be found in Figure 2.3.

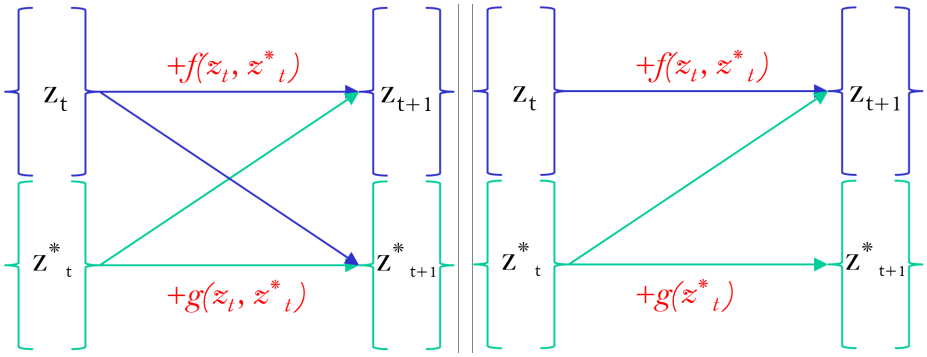

Fig. 2.3.: Architecture of discretized augmented neural ODEs (left) and of the discretized augmented neural ODE flows implemented in AFFJORD (right).

2.3 Proposed Framework

In the first subsection, we introduce our model AFFJORD, which is a special case of augmented neural ODEs. In the second subsection, we give the generalisation of the chain rule in the continuous sense which we refer to as the cable rule, which is analogous to forward sensitivity. Next, we give an intuitive proof of the continuous backpropagation, by showing its equivalence to the continuous generalisation of the total derivative decomposition. Then, using the cable rule, we give a more detailed explanation on why the instantaneous change of variable still holds in our model. In the final subsection, the augmented multiscale architecture is described, as it is implemented in the experimental section.

2.3.1 Proposed Model: Augmented FFJORD (AFFJORD)

A neural ODE of the type $f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})$, where $\boldsymbol{z}^*(t) = \boldsymbol{z}^*(t) + \int_0^t g(\boldsymbol{z}^*(\tau), \boldsymbol{\phi})d\tau$, can equivalently be written as $h_f(\boldsymbol{z}(t), t, \boldsymbol{\theta}) = f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})$. Thus, motivated by [Dupont, 2019], we propose to lift each data point $\boldsymbol{z}(0) \in \mathbb{R}^n$ to a higher dimensional space $\mathbb{R}^{n+m}$, by augmentation via an m dimensional vector $\boldsymbol{z}^*(0)$, which in practice is set to be the zero vector. Therefore, the joint vector $[\boldsymbol{z}(0), \boldsymbol{z}^*(0)]$ is transformed by the vector field $h = (f, g)$:

$$w(T) = \begin{bmatrix} \boldsymbol{z}(T) \\ \boldsymbol{z}^*(T) \end{bmatrix} = \begin{bmatrix} \boldsymbol{z}(0) \\ \boldsymbol{z}^*(0) \end{bmatrix} + \int_0^T \begin{bmatrix} f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta}) \\ g(\boldsymbol{z}^*(t), \boldsymbol{\phi}) \end{bmatrix} dt. \tag{2.8}$$

We are not interested in $z^*(T)$ as our interest lies on the transformed $z(0)$, that is $z(T)$. Since by definition such coupled dynamics are contained in the formulation of neural ODEs, this implies that the instantaneous change of formula still holds, and the transformation is injective. This sort of augmentation can be seen as a special case of augmentation introduced in [Dupont, 2019], where the augmented dimensions depend only on themselves. The augmented dimensions $\boldsymbol{z}^*(t)$ can also be seen as time dependent weights of the non-autonomous f whose evolution is determined by the autonomous ODE g, hence giving f greater flexibility in time [Zhang, 2019].

2.3.2 The Cable Rule

If we define a chain of transformations

$$\boldsymbol{z}_i = g_i(\boldsymbol{z}_{i-1}) \text{ for } i \in \{1, ..., n\}, \tag{2.9}$$

due to the chain rule we have:

$$\frac{d\boldsymbol{z}_n}{d\boldsymbol{z}_0} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}_{n-1}} \frac{d\boldsymbol{z}_{n-1}}{d\boldsymbol{z}_0} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}_{n-1}} \frac{\partial \boldsymbol{z}_{n-1}}{\partial \boldsymbol{z}_{n-2}} \cdots \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{z}_0} =$$

$$= \frac{\partial g_n(\boldsymbol{z}_{n-1})}{\partial \boldsymbol{z}_{n-1}} \frac{\partial g_{n-1}(\boldsymbol{z}_{n-2})}{\partial \boldsymbol{z}_{n-2}} \dots \frac{\partial g_1(\boldsymbol{z}_0)}{\partial \boldsymbol{z}_0}. \tag{2.10}$$

We can choose each g_i to infinitesimally modify its input, i.e., $g_i(\boldsymbol{z}_{i-1}) = \boldsymbol{z}_{i-1} + \epsilon f_i(\boldsymbol{z}_{i-1}, t_{i-1}, \boldsymbol{\theta})$, so that the chain in Expression 2.9 transforms $\boldsymbol{z}_0$ continuously. It is clear that $\boldsymbol{z}(t) = \boldsymbol{z}(0) + \int_0^t f(\boldsymbol{z}(t), t, \boldsymbol{\theta}) d\tau$ is the limit of the previous iterative definition when $\epsilon \to 0$. Then the expression $\frac{d\boldsymbol{z}_n}{d\boldsymbol{z}_0}$ in Equation (2.10) converges to $\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}$, which as we show in Appendix A.1, satisfies the differential equation below:

$$\frac{d(\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)})}{dt} = \frac{\partial f(\boldsymbol{z}(t))}{\partial \boldsymbol{z}(t)} \frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}. \tag{2.11}$$

Using the Magnus expansion [Magnus, 1954; Blanes, 2009], we conclude that

$$\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)} = e^{\boldsymbol{\Omega}(T)}, \text{ for } \boldsymbol{\Omega}(t) = \sum_{k=1}^{\infty} \boldsymbol{\Omega}_k(t), \tag{2.12}$$

where $\boldsymbol{\Omega}_i(t)$ are the terms of the Magnus expansion (See Appendix A.1). As this expression gives the generalisation of the chain rule in the continuous sense, we refer to it as the cable rule. We notice that Equation (2.11) gives the dynamics of the Jacobian of the state with respect to the initial condition $\boldsymbol{z(0)}$. The cable rule is therefore analogous to the forward sensitivity formula for ODEs which provides the dynamics of Jacobian of the state with respect to the parameters $\boldsymbol{\theta}$ of the flow [Zhao, 2013]. This relation is highlighted and explained in more detail in Appendix A.1. On this note, in Appendix A.7, we derive the cable rule via Equation (2.5). Furthermore, in Appendix A.5, we derive the instantaneous change of variables from the cable rule.

2.3.3 The Continuous Generalisation of the Total Derivative Decomposition and Continuous Backpropagation

In the case that $f = f(\boldsymbol{x}(\boldsymbol{\theta}), \boldsymbol{y}(\boldsymbol{\theta}))$, then $\frac{df}{d\boldsymbol{\theta}} = \frac{\partial f}{\partial \boldsymbol{x}} \frac{d\boldsymbol{x}}{d\boldsymbol{\theta}} + \frac{\partial f}{\partial \boldsymbol{y}} \frac{d\boldsymbol{y}}{d\boldsymbol{\theta}}$, as $\boldsymbol{\theta}$ contributes to both $\boldsymbol{x}$ and $\boldsymbol{y}$, which in turn determine f. If $\boldsymbol{z}(T) = \boldsymbol{z}(0) + \int_0^T f(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t) dt$, then $\boldsymbol{\theta}$ controls the vector field at each time point during integration, hence we expect that these infinitesimal contributions of the transformation from $\boldsymbol{z}(0)$ to $\boldsymbol{z}(T)$ should be integrated. Indeed, as we prove in Appendix A.2, the following holds:

$$\frac{d\boldsymbol{z}(T)}{d\boldsymbol{\theta}} = \int_0^T \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}(t)} \frac{\partial \boldsymbol{\theta}(t)}{\partial \boldsymbol{\theta}} dt, \tag{2.13}$$

from which for $\boldsymbol{\theta}(t) = \boldsymbol{\theta}$, and for some function $L = L(\boldsymbol{z}(T))$, we deduce:

$$\frac{dL}{d\boldsymbol{\theta}} = \frac{\partial L}{\partial \boldsymbol{z}(T)} \frac{d\boldsymbol{z}(T)}{d\boldsymbol{\theta}} = \int_0^T \frac{\partial L}{\partial \boldsymbol{z}(T)} \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} dt = \int_0^T \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} dt \tag{2.14}$$

thus giving an alternative and intuitive proof of continuous backpropagation [Pontrjagin, 1962; Chen, 2018]. In Appendix A.8, we show that the adjoint method can be used to prove Equation (2.13), hence the continuous total derivative decomposition is equivalent to continuous backpropagation. An alternate derivation of Equation (2.13) can be found in the Appendix of [Massaroli, 2020].

2.3.4 AFFJORD as a Special Case of Augmented Neural ODE Flows

As discussed in subsection 2.3.1, the ODE dynamics in Equation (2.8), can be seen as a special case of the following joint ODE transformation:

$$w(T) = \begin{bmatrix} \boldsymbol{z}(T) \\ \boldsymbol{z}^*(T) \end{bmatrix} = \begin{bmatrix} \boldsymbol{z}(0) \\ \boldsymbol{z}^*(0) \end{bmatrix} + \int_0^T \begin{bmatrix} f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta}) \\ g(\boldsymbol{z}^*(t), \boldsymbol{z}(t), \boldsymbol{\phi}) \end{bmatrix} dt. \tag{2.15}$$

In this general form, the model is unsuitable to be used in practice for two reasons:
1) The transformation of $\boldsymbol{z}(0)$ to $\boldsymbol{z}(T)$ is not necessarily injective. Indeed, the transformation from $[\boldsymbol{z}(0), \boldsymbol{z}^*(0)]$ to $[\boldsymbol{z}(T), \boldsymbol{z}^*(T)]$ is injective due to the Picard–Lindelöf Theorem, however, for two data points $\boldsymbol{z}'(0)$ and $\boldsymbol{z}''(0)$, their images $\boldsymbol{z}'(T)$, $\boldsymbol{z}''(T)$ might be identical as long as their respective augmented dimensions $\boldsymbol{z}'^*(T)$, $\boldsymbol{z}''^*(T)$ differ.
2) The Jacobian determinant of this general transformation is computationally intractable. Using the chain rule we can express the Jacobian determinant of this transformation as

$$\left|\det\left(\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)}\right)\right| = \left|\det\left(\frac{d\boldsymbol{z}(T)}{d\left[\boldsymbol{z}(T), \boldsymbol{z}^*(T)\right]} \frac{d\left[\boldsymbol{z}(T), \boldsymbol{z}^*(T)\right]}{d\left[\boldsymbol{z}(0), \boldsymbol{z}^*(0)\right]} \frac{d\left[\boldsymbol{z}(0), \boldsymbol{z}^*(0)\right]}{d\boldsymbol{z}(0)}\right)\right|. \tag{2.16}$$

The middle term on the RHS of Equation (2.16) can be further developed via the cable rule, to give the expression of the determinant of the Jacobian of augmented neural ODE flows, which is not computationally feasible in general.
As explained in Section 2.3.1, the special case (AFFJORD) formulated in Equation (2.8), mitigates the issues mentioned above.
Regarding issue 1), we have the following:

Proposition 2.3.1. *The architecture of AFFJORD ensures that the transformation is injective.*

Proof. Indeed, as $\boldsymbol{z}^*(t)$ is not dependent on external factors, and since $\boldsymbol{z}^*(0)$ is constant regarding $\boldsymbol{z}(0)$, the end result $\boldsymbol{z}^*(T)$ will always be the same. Hence, for $\boldsymbol{z}'(0)$ and $\boldsymbol{z}''(0)$ their images $\boldsymbol{z}'(T)$ and $\boldsymbol{z}''(T)$ must be different, since their equality would imply $[\boldsymbol{z}'(T), \boldsymbol{z}^*(T)] = [\boldsymbol{z}''(T), \boldsymbol{z}^*(T)]$ contradicting the Picard–Lindelöf Theorem. □
Issue 2) is mitigated as Equation (2.16) simplifies to

$$\left|\det\left(\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)}\right)\right| = \left|\det\left(\left[I, 0\right] \begin{bmatrix} e^{\int_0^T \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} dt + \boldsymbol{\Omega}_2^{[z]}(T) + \dots} & \bar{\boldsymbol{B}}(T) \\ 0 & \bar{\boldsymbol{D}}(T) \end{bmatrix} \begin{bmatrix} I \\ 0 \end{bmatrix}\right)\right|, \tag{2.17}$$

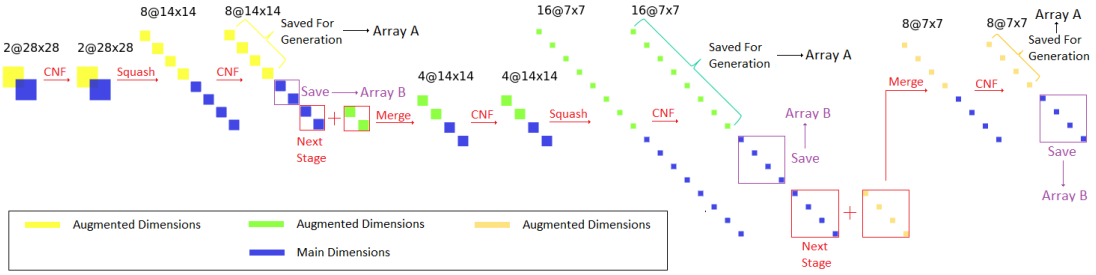

Fig. 2.4.: Example of the augmented multiscale architecture on MNIST dataset

for two block matrices $\bar{\boldsymbol{B}}(T)$ and $\bar{\boldsymbol{D}}(T)$. This is proven in Appendix A.6. In this case, the $[z]$ in $\boldsymbol{\Omega}_2^{[z]}(T)$ denotes the restriction of the Magnus expansion to the original dimensions. In case that the base distribution is multivariate normal, from

$$-\log p(\boldsymbol{z}(0)) = -\log \left[p(\boldsymbol{z}(T)) \left|\det\left(e^{\int_0^T \frac{\partial f(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} dt + \ldots}\right)\right|\right]$$

we derive the following loss function:

$$L = \frac{||\boldsymbol{Z}(T)||^2}{2} - \int_0^T tr \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t))}{\partial \boldsymbol{z}(t)} dt. \tag{2.18}$$

2.3.5 Multiscale Architecture in Augmented Neural ODE Flows

With reference to Figure 2.4, the multiscale architecture in the augmented case performs an Augmented Continuous Flow on the data as described in Subsection 2.3.1, then squeezes the channels (the augmented as well as the original channels) as in the original multiscale architecture. However after the second transformation the augmented channels are removed and stored in a separated array (Array A). The data channels are treated as before, that is, half of them are saved (Array B), and the other half are squeezed again. After this process is finished we add new augmented channels, to repeat the cycle. Note that in order to generate data by the inverse transformation, we need to retrieve the transformed augmented dimensions previously stored (i.e., Array A).

2.4 Experiments

We compare the performance of AFFJORD with respect to the base FFJORD on 2D data of toy distributions, as well as on standard benchmark datasets such as MNIST, CIFAR-10 and CelebA (32×32). In the case of the 2D data, we use the implementation of CNFs provided in [Chen, 2018], since using the Hutchinson's trace estimator and GPUs as in FFJORD provides no computational benefits in low dimensions. In this case we also use the non-adaptive Runge-Kutta 4 ODE solver, while for image data we use Dopri5, as well

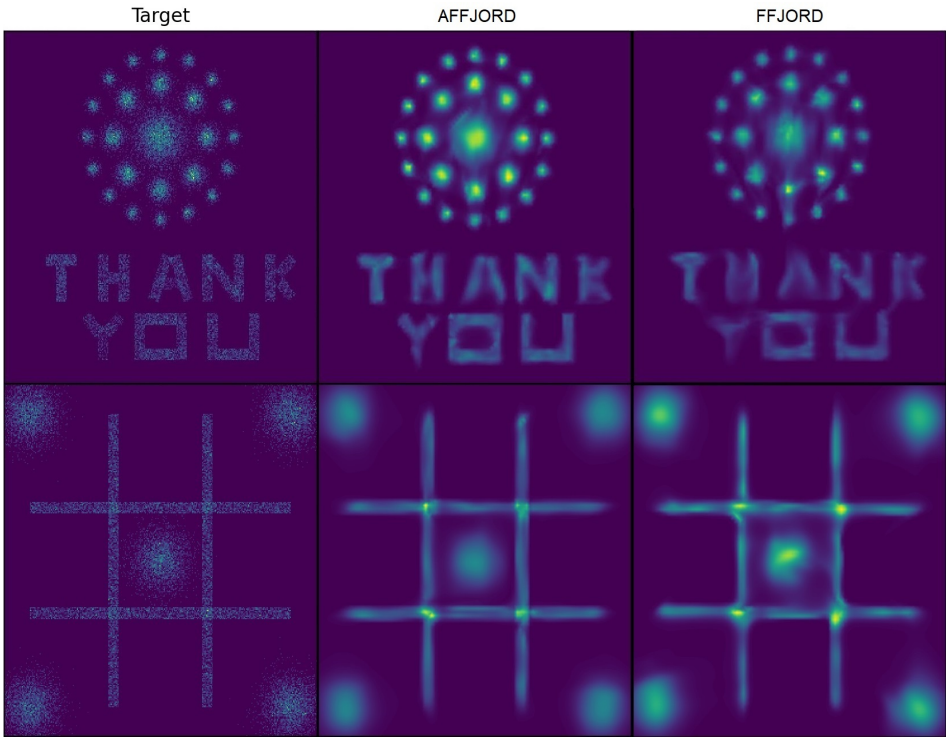

Fig. 2.5.: Probability density modeling capabilities of AFFJORD (second column) and FFJORD (third column) on 2D data of toy distributions.

as the FFJORD implementation of [Grathwohl, 2019]. We use a batch size of 200 for image data and a batch size of 512 for the toy datasets. In the case of image data, we use a learning rate of 6×10^{-4}, while for toy data the learning rate is set to 10^{-3}. All experiments were performed on a single GPU.

2.4.1 Toy 2D Datasets

In order to visualise the performance of the model, we first test FFJORD and AFFJORD on 2D data of toy distributions, depicted in Figure 2.5. In both cases we use the Runge-Kutta 4 solver with 40 time steps (160 function evaluations). In these examples, we used a hypernet architecture, where the augmented dimensions were fed to a hypernet (denoted as hyp) in order to generate the weights of the field of the main dimensions. The expression of the vector field to be learnt is the following: $\frac{dz}{dt} = f([\boldsymbol{z}(t), \boldsymbol{z}^*(t)], \boldsymbol{\theta}(t) = hyp(\boldsymbol{z}^*(t), \boldsymbol{w}))$, where $\frac{d\boldsymbol{z}^*(t)}{dt} = g(\boldsymbol{z}^*(t), \boldsymbol{\phi})$. Thus, the learnable parameters are $\boldsymbol{w}$ and $\boldsymbol{\phi}$.

While both FFJORD and AFFJORD are capable of modelling multi-modal and discontinuous distributions, Figure 2.5 shows that AFFJORD has higher flexibility in modeling the complex data distributions considered, in comparison to FFJORD. In the first row, the target is the TY distribution, where the datapoints form letters and a cluster of

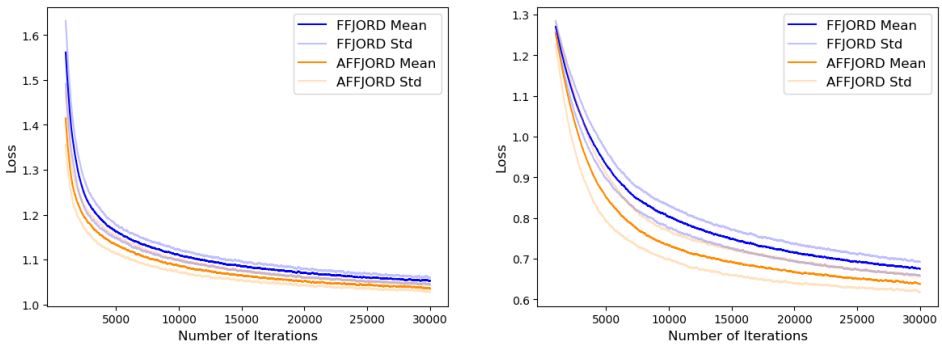

Fig. 2.6.: Performance of FFJORD and AFFJORD on the Hash-Gaussian toy 2D dataset (left) and TY toy 2D dataset (right).

Gaussian distributions. AFFJORD is more capable of separating the Gaussian spheres and modelling the shape of the text.

On the second row AFFJORD is capable of separating the Gaussian distribution in the center from the square that surrounds it. Furthermore, it separates the Gaussians in the corners from the hash symbol properly. In Figure 2.6, we show the results of the validation loss per iteration for both models. We can notice that the loss of our model is roughly two standard deviations lower than the one of FFJORD. For each model the experiment was repeated 30 times. The experiments provided here show that AFFJORD is characterized by high flexibility of the vector field. Indeed, in FFJORD the vector field changes more slowly, whereas in AFFJORD the field is able to change almost abruptly, due to this greater flexibility in time [1]. It is important to emphasize that AFFJORD retains the ability to generate samples from the learnt distribution, by simply integrating in the opposite direction. Indeed, for a given sample $\boldsymbol{z}_s(T)$ from the base distribution, we augment it to $\boldsymbol{z}^*(T)$, as $\boldsymbol{z}^*(T)$ is the same for all data points. Then, we can simply integrate backwards this concatenated vector $[\boldsymbol{z}_s(T), \boldsymbol{z}^*(T)]$ to $[\boldsymbol{z}_s(0), \boldsymbol{z}^*(0)]$, drop the generated augmented dimensions $\boldsymbol{z}^*(0)$, and simply keep the generated data point $\boldsymbol{z}_s(0)$.

2.4.2 Image Datasets

We show that AFFJORD outperforms FFJORD on MNIST, CIFAR-10 and CelebA (32×32). There are several architectures of FFJORD that can be used for this application. Out of the architectures that we tested, the one that performed best was the multiscale one, with three convolutional layers with 64 channels each. The number of CNF blocks was 1, and time was implemented by simply concatenating it as a channel into the data.

[1] Videos showing the comparison of dynamics between FFJORD and AFFJORD can be found at https://imgur.com/gallery/kMGCKve

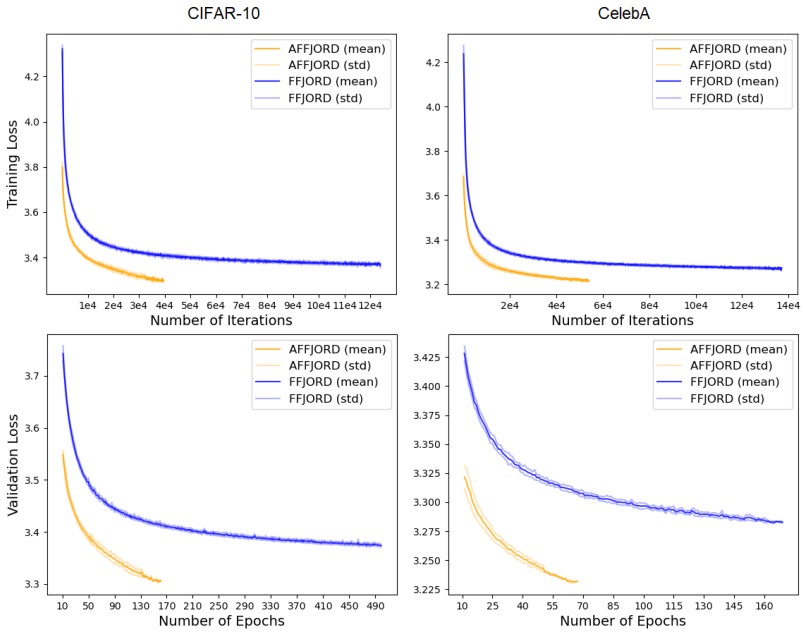

Fig. 2.7.: Graphs of training and evaluation losses of FFJORD and AFFJORD on CIFAR-10 as well as CelebA (32×32). We run the experiments 5 times. Lower is better.

When time was implemented via a hypernet, we observed that the training time increased and performance decreased, especially in the case of CIFAR-10. For AFFJORD we use the exact same base architecture, however, as in the case of 2D toy data, we enable the evolution of the vector fields through time via a hypernet which takes the augmented dimensions as an input, and outputs the weights of the main component. The augmented dimensions are concatenated as a channel to the main channel of the data. The formulas for both the main field and the augmented component remain unchanged from the case of the 2D toy data, that is, $\frac{dz}{dt} = f([\boldsymbol{z}(t), \boldsymbol{z}^*(t)], \boldsymbol{\theta}(t) = hyp(\boldsymbol{z}^*(t), \boldsymbol{w}))$ and $\frac{d\boldsymbol{z}^*(t)}{dt} = g(\boldsymbol{z}^*(t), \boldsymbol{\phi})$.

The main difference here is that $g(\boldsymbol{z}^*(t), \boldsymbol{\phi})$ is a fully connected network with one hidden layer, which does not take as input all the dimensions of $\boldsymbol{z}^*(t)$ but merely 20 of them. Number 20 was chosen during fine-tuning, as the best performance was reached in this setting. The width of the hidden layer is also 20, as is the output. We fix 10 of these 20 dimensions and feed them to a linear hypernet with weight matrix shape $[10, p]$ to output

Tab. 2.1.: Experimental Results for Density Estimation Models, in Bits/Dim for MNIST, CIFAR-10 and CelebA (32×32). Lower Is Better. The Multiscale Architecture Is Used in All Cases.

MODEL	MNIST	CIFAR10	CelebA(32×32)
Real NVP	1.06	3.49	-
Glow	1.05	3.35	-
FFJORD	0.96 $\pm$.00	3.37 $\pm$.00	3.28$\pm$.00
AFFJORD	**0.95** $\pm$ .01	**3.32** $\pm$ .01	**3.23** $\pm$ .00

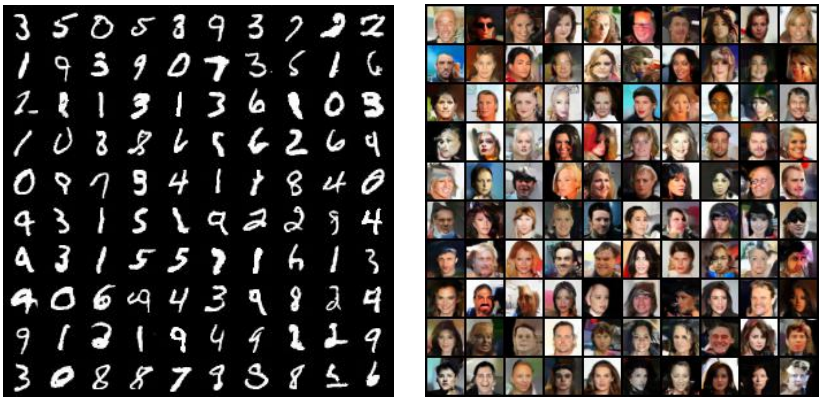

Fig. 2.8.: Samples generated from AFFJORD: MNIST and CelebA 32×32.

the p weights for the main component. It should be emphasized that the architecture of FFJORD in the main dimensions remains unchanged in AFFJORD for fair comparison, and we only fine-tuned the augmented structure in addition. Additional details about the experimental settings can be found in Appendix A.9.

As we show in Table 2.1, AFFJORD slightly outperforms FFJORD on MNIST on the best run, since both models reach optimal performance, as seen from the generated samples in Figure 2.8. However, our model outperforms FFJORD on the CIFAR-10 and CelebA (32×32) dataset, as illustrated in Table 2.1, as well as in Figure 2.7. Based on the conducted experiments the farther FFJORD is from optimal performance, the larger the improvements brought by AFFJORD are. The calculation of results is done as in [Grathwohl, 2019], where for each run the best evaluation result over epochs is taken. After 5 runs, for each model, the scores are averaged and reported in Table 2.1.

In the case of MNIST, both models were trained for roughly 9 days, while in the case of CIFAR-10 and CelebA (32×32), they were trained for approximately 14 days. The results corresponding to the Real NVP and Glow models, are taken from the original papers: [Dinh, 2017] and [Kingma, 2018].

As in the previous case, AFFJORD can generate samples by backintegrating. However, due to the use of the augmented multiscale architecture, where for each cycle we replace the augmented dimensions, these replaced augmented dimensions must be saved in an array for the backward generative pass. Examples of samples from AFFJORD are shown in Figure 2.8 for both MNIST and CelebA (32×32) datasets. Additional generated samples from both AFFJORD and FFJORD can be found in Appendix A.4.

2.5 Limitations and Future Work

Number of Function Evaluations (NFE)

As originally reported in [Grathwohl, 2019], the number of function evaluations is one of the main bottlenecks of Neural ODE-based models. Interestingly, if the concatenation architecture is used in AFFJORD, that is, we only replace the concatenated time channel in FFJORD with the augmented channel in AFFJORD, then the number of function evaluations decreases significantly. This aspect is illustrated in Figure 2.9. However, the hypernet architecture of AFFJORD is prone to the issue of the number of function evaluations. Indeed, forward passes in AFFJORD-hypernet can be challenging, as the learnt vector field becomes too stiff with an increasing number of integration steps.

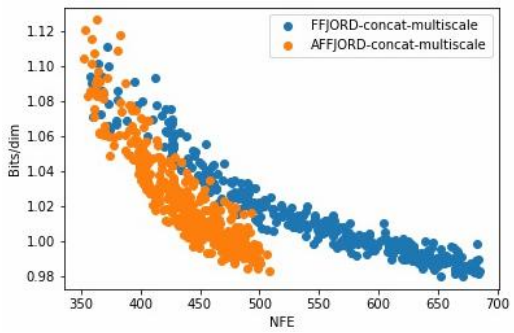

Fig. 2.9.: Number of function evaluations of FFJORD and AFFJORD when the 'concatenate' architecture is used.

Addition of Self-Attention

[Ho, 2019] describe three modeling inefficiencies in prior work on flow models. One such factor is the lack of expressive power of convolutional layers used in normalizing flow models. Considering the performance improvements demonstrated in [Ho, 2019], in the future we intend to test the improvement in performance brought by the addition of self-attention in AFFJORD.

Data Dependent Augmented Dimensions

As described in Section 2.3.1, several simplifications are made in the architecture of the general augmented neural ODE flows, in order to ensure immediate bijectivity and reduce the computational complexity. However, other possible architectures exist, where for example $\boldsymbol{z}^*(0)$ is dependent on $\boldsymbol{z}(0)$, and $g(\boldsymbol{z}^*(t)) = -\boldsymbol{z}^*(t)$. This would ensure that the augmented dimensions converge to zero, providing bijectivity. Since all augmented dimensions would be different during training, this would imply that the data is lifted to a higher plane, enabling richer transformations. Finding an approximation of the loss in

Equation (2.16) remains a challenge for the future however.

Jacobian Regularization

Theoretically speaking, all performance enhancing modifications that can be applied to FFJORD are also applicable to AFFJORD. Such a modification which reduces the training time of FFJORD is presented in [Finlay, 2020], where both the vector field and its Jacobian are regularized. Thus, an interesting research direction in the future would be to test how the performance of AFFJORD is affected by such amendments.

2.6 Conclusion

We have presented the generalization of the total derivative decomposition in the continuous sense as well as the continuous generalization of the chain rule, to which we refer as the cable rule. The cable rule is analogous to the forward sensitivity of ODEs in the sense that, it gives the dynamics of the Jacobian of the state with respect to the initial conditions, whereas forward sensitivity gives the dynamics of the Jacobian of the state with respect to the parameters of the flow. Motivated by this contribution, we propose a new type of continuous normalizing flow, namely Augmented FFJORD (AFFJORD), which outperforms the CNF state-of-art-approach, FFJORD, in the experiments we conducted on the task of density estimation on both 2D toy data, and on high dimensional datasets such as MNIST, CIFAR-10 and CelebA (32×32).

3

Faster Training of Diffusion Models and Improved Density Estimation via Parallel Score Matching

Contents

As elucidated in our introduction, in Diffusion Probabilistic Models (DPMs), the task of modeling the score evolution via a single time-dependent neural network necessitates extended training periods and may potentially impede modeling flexibility and capacity. To counteract these challenges, in this chapter, we propose leveraging the independence of learning tasks at different time points inherent to DPMs. More specifically, we partition the learning task by utilizing independent networks, each dedicated to learning the evolution of scores within a specific time sub-interval. Further, inspired by residual flows, we extend this strategy to its logical conclusion by employing separate networks to independently model the score at each individual time point. As empirically demonstrated on synthetic and image datasets, our approach not only significantly accelerates the training process by introducing an additional layer of parallelization atop data parallelization, but it also enhances density estimation performance when compared to the conventional training methodology for DPMs.

3.1 Introduction

Forward Diffusion Processes (FDPs) represent a category of Markov chains that methodically metamorphose the data distribution into a standard multivariate normal one by incrementally corrupting the data samples. Within the scope of Diffusion Probabilistic Models (DPMs), the machine learning task encompasses the training of a neural network with the objective of emulating the reverse dynamics of this process [Sohl-Dickstein, 2015]. To this end, different approaches have been developed, notably the denoising [Vincent, 2011; Ho, 2020] and the sliced score matching loss [Hyvärinen, 2005; Song, 2020]. In both cases, the observed samples are used to train a function approximator in order to model the score, that is, the gradient field of the log-likelihood of the probability density function (pdf) from which the samples originate.

The performance of such models improves as the number of steps in the FDP increases. The optimum is naturally reached at the limit when the number of steps tends to infinity. In this case, the FDP can be depicted as a Stochastic Differential Equation (SDE). This SDE defines a distinct distribution at every temporal point of the diffusion process, with the initial and terminal ones being the data and the standard Gaussian distributions respectively. Given samples from the data distribution, it is possible to efficiently generate samples from the distribution at any specified time t_i. These generated samples can then be utilized to train a neural network to approximate the score at time t_i. The common approach [Ho, 2020; Song, 2021; Rombach, 2022] is to train a single time-varying neural network to model all scores, i.e., to model the evolution of the score (Figure 3.1).

The Fokker-Planck equation establishes the connection between SDE diffusion models and continuous normalizing flows [Song, 2021]. Indeed, the estimated scores provide the dynamics of the continuous normalizing flow (CNF) which describes the evolution of the distribution determined by the SDE. Leveraging on this connection, the CNF framework can be deployed to generate data, or to estimate the likelihood of unobserved points through the application of the instantaneous change of variable.

Despite the inherent equivalence and functionality between SDE diffusion processes and CNFs, their optimization techniques exhibit differences. CNFs are trained by maximizing likelihood, a methodology that is suboptimal for numerous reasons. A key reason is that normalizing flows, being designed as a sequence of transformations, necessitate the retention of the entire sequence in memory during training, as it is infeasible to optimize a step in the sequence independently from the rest. While CNFs alleviate the memory bottleneck via the adjoint method, the continuity requirement dictates that all vector fields at all time points must be learned by a singular time-dependent neural network, which impinges upon the transformation's flexibility. The composition of multiple CNFs together is rendered impracticable due to memory and computational constraints, as

the sequential nature of the composition, as before, implies that each additional CNF introduces a new set of parameters and extends the training duration. In this context, each CNF is referred to as a CNF block.

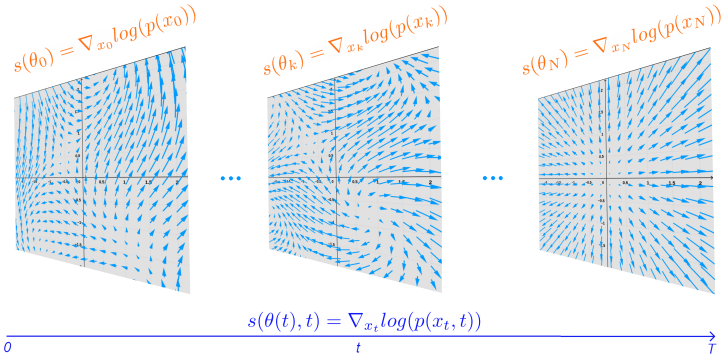

Fig. 3.1.: Instead of using a time dependent network to learn the scores for all times (below: blue), at each time point we can use a model per score (above: orange), naturally allowing parallel training of diffusion models.

The aforementioned points underscore the impracticability of independent learning of vector fields in the CNF approach. To expedite training and augment model flexibility, we leverage the intrinsic property of DPMs that allows scores at different time points to be optimized independently. Consequently, instead of training a single time-varying U-Net to model the scores of the distributions at all time points, we partition the training task by dividing the integration time interval of the SDE into smaller sub-intervals. For each sub-interval, we employ a time-dependent U-Net to model the evolution of scores for the distributions defined within that sub-interval. This strategy is referred to as time-varying parallel score matching (TPSM). Each such U-Net corresponds to a CNF block within the CNF framework, and their composition provides the complete evolution of distribution as determined by the FDP.

With an increasing number of such sub-intervals, the task assigned to each U-Net is simplified, and in the limit, each time sub-interval converges to a point. Consequently, in addition to the previously described strategy, we train a multitude of smaller networks, which are not time-dependent, such that each network learns the score at a single time point of the diffusion process (Figure 3.1). This methodology is designated as Discrete Parallel Score Matching (DPSM).

We evaluate our Parallel Score Matching (PSM) methodologies on 2D data, in addition to image datasets such as CIFAR-10, CelebA 64×64, and ImageNet 64×64, [Liu, 2015; Deng, 2009]. The findings attest that our approach not only enhances the log-likelihood results, but also expedites the training process, without incurring penalties related to memory or inference time.

3.2 Background and Related Work

3.2.1 Normalizing Flows (NFs)

Normalizing Flows: A Normalizing Flow [Dupont, 2019; Tabak, 2013; Rezende, 2015; Dinh, 2015] is a transformation defined as a sequence of diffeomorphisms that converts a base probability distribution (e.g., a standard normal) into another distribution by warping the domain on which they are defined. Let $\boldsymbol{Z}$ be a random variable, and define $\boldsymbol{X} = g(\boldsymbol{Z})$, where g is a diffeomorphism with inverse h. If we denote their probability density functions by $f_{\boldsymbol{Z}}$ and $f_{\boldsymbol{X}}$, the change of variable theorem states that:

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = f_{\boldsymbol{Z}}(\boldsymbol{z}) \left| \det \left(\frac{d\boldsymbol{z}}{d\boldsymbol{x}} \right) \right| = f_{\boldsymbol{Z}}(h(\boldsymbol{x})) \left| \det \left(\frac{dh(\boldsymbol{x})}{d\boldsymbol{x}} \right) \right|. \tag{3.1}$$

The objective is the optimization of parameters of h to maximize the likelihood of sampled data points $\boldsymbol{x}$. After training, one can input any test point $\boldsymbol{x}$ on the RHS of Equation (3.1), and calculate its likelihood. For the generative task, being able to easily recover g from h is essential, as the generated point $\boldsymbol{x}_g$ will take form $\boldsymbol{x}_g{=}g(\boldsymbol{z}_s)$, where $\boldsymbol{z}_s$ is a sampled point from the base distribution $f_{\boldsymbol{Z}}$.
For increased modeling flexibility, we can use a chain (flow) of transformations, $\boldsymbol{x}_{i-1} = g_i(\boldsymbol{x}_i),\ i \in [n]$, that is $\boldsymbol{x}_i = h_i(\boldsymbol{x}_{i-1}),\ i \in [n]$. In this case, due to chain rule we have:

$$f_{\boldsymbol{X}_0}(\boldsymbol{x}_0) = f_{\boldsymbol{X}_n}(\boldsymbol{x}_n) \left| \det \left(\frac{d\boldsymbol{x}_n}{d\boldsymbol{x}_0} \right) \right| = f_{\boldsymbol{X}_n}(h_n(...h_1(\boldsymbol{x}_0))) \prod_{i=1}^{n} \left| \det \left(\frac{dh_i(\boldsymbol{x}_{i-1})}{d\boldsymbol{x}_{i-1}} \right) \right|, \tag{3.2}$$

where $\boldsymbol{x}_0$ is a data point. The interested reader can find a more in-depth review of normalizing flows in [Kobyzev, 2021] and [Papamakarios, 2021].

Neural ODE Flows: Neural ODEs [Chen, 2018] are continuous generalizations of residual networks:

$$\begin{aligned} \boldsymbol{x}_{t_{i+1}} &= \boldsymbol{x}_{t_i} + \epsilon f(\boldsymbol{x}_{t_i}, t_i, \boldsymbol{\theta}) \\ \rightarrow \boldsymbol{x}(t) &= \boldsymbol{x}(0) + \int_0^t f(\boldsymbol{x}(\tau), \tau, \boldsymbol{\theta}) d\tau, \text{ as } \epsilon \rightarrow 0, \end{aligned} \tag{3.3}$$

where for a network with finite weights, injectivity is guaranteed by the Picard–Lindelöf theorem. In [Chen, 2018], the expression for the instantaneous change of variable is derived, which enables one to train continuous normalizing flows and perform likelihood estimation:

$$\log p(\boldsymbol{x}(0)) = \log p(\boldsymbol{x}(T)) + \int_0^T tr \frac{\partial f(\boldsymbol{x}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{x}(t)} dt, \tag{3.4}$$

where $\boldsymbol{x}(0)$ represents a sample from the data. Such models are better known as Continuous Normalizing Flows (CNFs).

Invertible Residual Flows (IRFs): These models [Behrmann, 2019] are similar to the discretized version of Continuous Normalizing Flows:

$$\boldsymbol{x}_{t_{i+1}} = \boldsymbol{x}_{t_i} + f(\boldsymbol{x}_{t_i}, \boldsymbol{\theta_{t_i}})$$

It can be noticed that in this case the output of the residual block is not scaled before being added to the input, and furthermore, each block is comprised of a different set of trainable parameters, instead of allowing the behaviour of the model to evolve via a time input. As injectivity cannot be guaranteed via the Picard–Lindelöf theorem, bijectivity has to be enforced by imposing the contraction condition $Lip(f(\boldsymbol{x}_{t_i}, \boldsymbol{\theta_{t_i}})) < 1$, where $Lip(f(\boldsymbol{x}_{t_i}, \boldsymbol{\theta_{t_i}}))$ is the Lipschitz constant of $f(\boldsymbol{x}_{t_i}, \boldsymbol{\theta_{t_i}})$.

3.2.2 Diffusion Probabilistic Models (DPMs)

Denoising Diffusion Models: A forward diffusion process or diffusion process [Sohl-Dickstein, 2015] is a fixed Markov chain that gradually adds Gaussian noise to the data according to a schedule $\{b_t | t \in [n]\}$:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) := \mathcal{N}(\boldsymbol{x}_{t-1}\sqrt{1-b_t}, b_t\boldsymbol{I}). \tag{3.5}$$

Such a process transforms the data distribution into a standard multivariate normal distribution. If we denote $a_t = 1 - b_t$ and $\bar{a}_t = \prod_{s=1}^{t} a_s$, then we can write:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0\sqrt{\bar{a}_t}, (1-\bar{a}_t)\boldsymbol{I}). \tag{3.6}$$

The reverse process conditioned on the initial sample is also described by a chain of Gaussian distributions:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}(\mu(\boldsymbol{x}_t, \boldsymbol{x}_0), \boldsymbol{\Sigma}(t)), \tag{3.7}$$

where

$$\mu(\boldsymbol{x}_t, \boldsymbol{x}_0) = \mu(\boldsymbol{x}_t, \boldsymbol{x}_0(\boldsymbol{x}_t, \boldsymbol{\varepsilon})) = \frac{1}{\sqrt{a_t}}(\boldsymbol{x}_t - \frac{b_t}{\sqrt{1-\bar{a}_t}}\boldsymbol{\varepsilon}),$$
$$\boldsymbol{\Sigma}(t) = b_t\boldsymbol{I}.$$

The goal is to approximate this reverse process via

$$p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}(\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}(t)) \tag{3.8}$$

that converts the standard multivariate normal distribution into the data distribution. To this end, for each step t, the KL divergence between $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ and $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is minimized, which amounts to minimizing

$$\mathbb{E}_{\boldsymbol{x_0}, \boldsymbol{x_t}} ||\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \mu(\boldsymbol{x}_0, \boldsymbol{x}_t)||^2, \tag{3.9}$$

or equivalently

$$\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\varepsilon}}||\boldsymbol{\varepsilon}_\theta(\boldsymbol{x_0}\sqrt{\bar{a}_t}+\sqrt{(1-\bar{a}_t)}\boldsymbol{\varepsilon},t)-\boldsymbol{\varepsilon}||^2, \tag{3.10}$$

for $\boldsymbol{\varepsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})$, data samples $\boldsymbol{x_0}$, and a neural network $\boldsymbol{\varepsilon}_\theta(\boldsymbol{x_t},t)$ [Ho, 2020].

SDE Diffusion Models and their CNF Representation: For an $\boldsymbol{\varepsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})$, the forward diffusion process defined in Equation (3.5) can be written as

$$\boldsymbol{x}_t=\boldsymbol{x}_{t-1}\sqrt{1-b_t}+\sqrt{b_t}\boldsymbol{\varepsilon}. \tag{3.11}$$

As derived in [Song, 2021], the continuous counterpart of this process takes the form

$$d\boldsymbol{x}(t)=-\frac{1}{2}b(t)\boldsymbol{x}(t)dt+\sqrt{b(t)}d\boldsymbol{w}. \tag{3.12}$$

In this case Equation (3.6) becomes

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0)=\mathcal{N}(\boldsymbol{x}_0\mu_t,\sigma_t^2\boldsymbol{I}), \tag{3.13}$$

where $\mu_t=e^{-\frac{1}{2}\int_0^t b(s)ds}$ and $\sigma_t=\sqrt{(1-e^{-\int_0^t b(s)ds})}$.
As shown through the Fokker-Plack equation, the evolution of the probability density function of the data, as dictated by the FDP, is identical to the evolution dictated by the following ODE transformation:

$$d\boldsymbol{x}(t)=-\frac{1}{2}b(t)[\boldsymbol{x}(t)+\nabla_{\boldsymbol{x}(t)}\log p_t(\boldsymbol{x}(t))]dt=f_t(\boldsymbol{x}(t))dt. \tag{3.14}$$

Indeed, the Fokker-Planck equation of

$$d\boldsymbol{x}=h(\boldsymbol{x},t)dt+g(t)d\boldsymbol{w}=h(\boldsymbol{x},t)dt+g(t)Id\boldsymbol{w} \tag{3.15}$$

is

$$\frac{\partial p_t(\boldsymbol{x})}{\partial t}=-\sum_{i=1}^{d}\frac{\partial}{\partial \boldsymbol{x}_i}[h_i(\boldsymbol{x},t)p_t(\boldsymbol{x})]+\frac{1}{2}\sum_{i=1}^{d}\sum_{i=j}^{d}\frac{\partial^2}{\partial \boldsymbol{x}_i\partial \boldsymbol{x}_j}[g^2(t)Ip_t(\boldsymbol{x})]. \tag{3.16}$$

Since the entries ij, where $i\neq j$ of $g(t)I$ are zero we have:

$$\frac{\partial p_t(\boldsymbol{x})}{\partial t}=-\sum_{i=1}^{d}\frac{\partial}{\partial \boldsymbol{x}_i}[h_i(\boldsymbol{x},t)p_t(\boldsymbol{x})]+\frac{1}{2}\sum_{i=1}^{d}\frac{\partial^2}{\partial \boldsymbol{x}_i\partial \boldsymbol{x}_i}[g^2(t)p_t(\boldsymbol{x})], \tag{3.17}$$

$$\frac{\partial p_t(\boldsymbol{x})}{\partial t}=-\sum_{i=1}^{d}\frac{\partial}{\partial \boldsymbol{x}_i}[h_i(\boldsymbol{x},t)p_t(\boldsymbol{x})]+\frac{1}{2}\sum_{i=1}^{d}\frac{\partial}{\partial \boldsymbol{x}_i}[g^2(t)\frac{\partial p_t(\boldsymbol{x})}{\partial \boldsymbol{x}_i}], \tag{3.18}$$

$$\frac{\partial p_t(\boldsymbol{x})}{\partial t}=-\sum_{i=1}^{d}\frac{\partial}{\partial \boldsymbol{x}_i}[h_i(\boldsymbol{x},t)p_t(\boldsymbol{x})]+\frac{1}{2}\sum_{i=1}^{d}\frac{\partial}{\partial \boldsymbol{x}_i}[g^2(t)\frac{\partial \log p_t(\boldsymbol{x})}{\partial \boldsymbol{x}_i}p_t(\boldsymbol{x})], \tag{3.19}$$

$$\frac{\partial p_t(\boldsymbol{x})}{\partial t}=-\sum_{i=1}^{d}\frac{\partial}{\partial \boldsymbol{x}_i}[h_i(\boldsymbol{x},t)p_t(\boldsymbol{x})]+\frac{1}{2}\sum_{i=1}^{d}\frac{\partial}{\partial \boldsymbol{x}_i}[g^2(t)(\nabla_{\boldsymbol{x}}\log p_t(\boldsymbol{x}))_i p_t(\boldsymbol{x})], \tag{3.20}$$

$$\frac{\partial p_t(\boldsymbol{x})}{\partial t} = -\sum_{i=1}^{d} \frac{\partial}{\partial \boldsymbol{x}_i}[h_i(\boldsymbol{x},t)p_t(\boldsymbol{x}) - \frac{1}{2}g^2(t)(\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}))_i p_t(\boldsymbol{x})], \tag{3.21}$$

$$\frac{\partial p_t(\boldsymbol{x})}{\partial t} = -\sum_{i=1}^{d} \frac{\partial}{\partial \boldsymbol{x}_i}[(h_i(\boldsymbol{x},t) - \frac{1}{2}g^2(t)(\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}))_i)p_t(\boldsymbol{x})], \tag{3.22}$$

which is the Fokker-Planck equation of

$$d\boldsymbol{x}(t) = h(\boldsymbol{x},t)\ dt - \frac{1}{2}g^2(t)\nabla_{\boldsymbol{x}(t)} \log p(\boldsymbol{x}(t),t)dt. \tag{3.23}$$

Setting $h(\boldsymbol{x},t) = -\frac{1}{2}b(t)\boldsymbol{x}(t)$ and $g(t) = \sqrt{b(t)}$, we have

$$d\boldsymbol{x}(t) = -\frac{1}{2}b(t)[\boldsymbol{x}(t) + \nabla_{\boldsymbol{x}(t)} \log p_t(\boldsymbol{x}(t))]dt = f_t(\boldsymbol{x}(t))dt. \tag{3.24}$$

Knowing f_t allows us to perform data generation and likelihood estimation, through the framework of continuous normalizing flows. It can be observed that the only unknown in f_t, is the score $\nabla_{\boldsymbol{x}(t)} \log p_t(\boldsymbol{x}(t))$. This quantity can be modelled using a neural network $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}(t),t)$ trained either through sliced score matching [Song, 2020]:

$$min\ \mathbb{E}_{\boldsymbol{x}_t \sim p(\boldsymbol{x}_t)}[\frac{1}{2}||\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)||^2 + \text{div}(\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t))], \tag{3.25}$$

or the equivalent MSE denoising loss [Ho, 2020; Kingma, 2021]:

$$min\ \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\varepsilon}}||\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_0\mu_t + \sigma_t\boldsymbol{\varepsilon},t) - (-\frac{\boldsymbol{\varepsilon}}{\sigma_t})||^2. \tag{3.26}$$

One can prove that the last loss ensures our network $\boldsymbol{s}_{\boldsymbol{\theta}}$ learns the scores. First,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\varepsilon}}||s_{\boldsymbol{\theta}}(\boldsymbol{x}_0\mu_t + \sigma_t\boldsymbol{\varepsilon}) - (-\frac{\boldsymbol{\varepsilon}}{\sigma_t})||^2, \tag{3.27}$$

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{x}_t}||s_{\boldsymbol{\theta}}(\boldsymbol{x}_t) - (-\frac{\boldsymbol{x}_t - \boldsymbol{x}_0\mu_t}{\sigma_t^2})||^2, \tag{3.28}$$

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{x}_t}||s_{\boldsymbol{\theta}}(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}_0)||^2. \tag{3.29}$$

Second, we have

$$\mathbb{E}_{(\boldsymbol{x}_0,\boldsymbol{x}_t)\sim p(\boldsymbol{x}_0,\boldsymbol{x}_t)}||s_{\boldsymbol{\theta}}(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}_0)||^2 =$$

$$= \mathbb{E}_{\boldsymbol{x}_t \sim p(\boldsymbol{x}_t)}\mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}||s_{\boldsymbol{\theta}}(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}_0)||^2. \tag{3.30}$$

Then, we can develop the last expression even further as follows

$$\mathbb{E}_{\boldsymbol{x}_t \sim p(\boldsymbol{x}_t)}\mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}||s_{\boldsymbol{\theta}}(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}_0)||^2 =$$

$$\mathbb{E}_{\boldsymbol{x}_t \sim p(\boldsymbol{x}_t)}\mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}||s_{\boldsymbol{\theta}}(\boldsymbol{x}_t) - \mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}_0) +$$
$$+\mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)}\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}_0) - \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}_0)||^2$$

$$= C + \mathbb{E}_{\boldsymbol{x}_t \sim p(\boldsymbol{x}_t)} \mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)} ||s_{\boldsymbol{\theta}}(\boldsymbol{x}_t) - \mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)} \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}_0)||^2,$$

where C does not depend on $\boldsymbol{\theta}$. Thus,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}_t \sim p(\boldsymbol{x}_t)} ||s_{\boldsymbol{\theta}}(\boldsymbol{x}_t) - \mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)} \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}_0)||^2. \tag{3.31}$$

Finally, since

$$\mathbb{E}_{\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0|\boldsymbol{x}_t)} \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}_0) = \int \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}_0) p(\boldsymbol{x}_0|\boldsymbol{x}_t) d\boldsymbol{x}_0 =$$

$$= \frac{\int \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t, \boldsymbol{x}_0) p(\boldsymbol{x}_0, \boldsymbol{x}_t) d\boldsymbol{x}_0}{p(\boldsymbol{x}_t)} = \frac{\int \nabla_{\boldsymbol{x}_t} p(\boldsymbol{x}_0, \boldsymbol{x}_t) d\boldsymbol{x}_0}{p(\boldsymbol{x}_t)} = \frac{\nabla_{\boldsymbol{x}_t} \int p(\boldsymbol{x}_0, \boldsymbol{x}_t) d\boldsymbol{x}_0}{p(\boldsymbol{x}_t)} =$$

$$= \frac{\nabla_{\boldsymbol{x}_t} p(\boldsymbol{x}_t)}{p(\boldsymbol{x}_t)} = \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t),$$

we conclude that

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}_t \sim p(\boldsymbol{x}_t)} ||s_{\boldsymbol{\theta}}(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)||^2. \tag{3.32}$$

The facts proven above, motivate and justify Algorithm 1.

Algorithm 1 Standard Approach in Diffusion Models

Input: data $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$, batch-size p
Train model $\boldsymbol{s}_{\boldsymbol{\theta}}(t)$
repeat
  sample τ_1, ..., τ_p from $[0, 1]$
  sample $\boldsymbol{x}^0_{\pi(1)}$, ..., $\boldsymbol{x}^0_{\pi(p)}$ from $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$
  generate $\boldsymbol{x}^{\tau_1}_{\pi(1)}$, ..., $\boldsymbol{x}^{\tau_p}_{\pi(p)}$ using Equation (3.13)
  minimize $\Sigma_j ||\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}^{\tau_j}_{\pi(j)}(\boldsymbol{\varepsilon_j}), \tau_j) - (\boldsymbol{\varepsilon_j})||^2$.
until convergence

3.3 PSM Framework and Relation to Piece-wise Continuous Flows and IRFs

In the context of continuous normalizing flows, given the forward integration, the expression $\boldsymbol{x}(t) = \boldsymbol{x}(0) + \int_0^T f_t dt$ is comprised of all vector fields $f_t = f(\boldsymbol{x}(t), \boldsymbol{\theta}, t)$, parameterized by the network's trainable parameters, as well as the time input. Increasing model capacity by using a distinct neural network to learn a vector field at each time t is not feasible, as the entire function $\boldsymbol{x}(t_N) = \boldsymbol{x}(\boldsymbol{x}_0, f_{t_1}(\boldsymbol{\theta}_{t_1}), f_{t_2}(\boldsymbol{\theta}_{t_2}), ..., f_{t_N}(\boldsymbol{\theta}_{t_T}))$ would need to be maintained in memory, and the transformation may not necessarily be continuous. Memory constraints also surface when employing multiple CNF blocks within a piece-wise continuous flow, as each block $\int_{t_{i-1}}^{t_i} f(\boldsymbol{x}(\tau), \boldsymbol{\theta}_i, \tau) d\tau$ introduces a new

Tab. 3.1.: A Comparison of the Properties of Parallel Score Matching (PSM) Approaches with Other Generative Diffeomorphism-Based Frameworks.

Method:	CNF	SA-DPM	PSM-DPM
Computationally Demanding Loss	√	×	×
Memory Bottleneck	√	×	×
Limited Variability In Time	√	√	×
Necessity To Devise A diffeomorphism	√	×	×
Score Optimization Interdependency	√	√	×

set of parameters $\boldsymbol{\theta}_i$. Given that these transformations are interconnected in a chain, they must coexist in memory during the maximum likelihood optimization procedure. Moreover, continuous normalizing flows present a formidable optimization task, as the time-dependent network $f(\boldsymbol{x}(t), \boldsymbol{\theta}, t)$ is obligated to generate an evolution process for the data distribution that culminates in a standard normal distribution. The loss function further adds to the complexity, as not only it contains the end point of the integration path $\boldsymbol{x}(T) = \boldsymbol{x}(0) + \int_0^T f(\boldsymbol{x}(t), \boldsymbol{\theta}, t)dt$, but also the integral of the divergence along that integration path $\int_0^T tr\frac{\partial f(\boldsymbol{x}(t),t,\boldsymbol{\theta})}{\partial \boldsymbol{x}(t)}dt$.
Contrarily, diffusion models parameterized by time-dependent neural networks ameliorate the complexity of learning the evolution process. This is because the forward diffusion delineates the time-based evolution of the data distribution, simplifying the framework's task to merely modeling the already defined vector fields. However, if parameterized by a single time-varying network, these models still grapple with limited flexibility due to a single set of parameters (model) being tasked with modeling all the vector fields defined by the FDP. Moreover, the optimization of a vector field at time t is dependent on the optimization of other vector fields at various time points, which hampers the speed of the training process. This standard approach (SA) in Diffusion Probabilistic Models (DPMs) is outlined in Algorithm 1 for comparison, and is abbreviated as SA-DPM.

Disadvantages of SA in DPMs

1) The flow's adaptability over time is limited due to the employment of a single time-varying network.
2) The optimization of score approximation at any given time t_i remains influenced by the optimization of score approximation at any other time t_j.

As shown in Appendix B.4, the scores in a diffusion process evolve continuously, with the given PDE dynamics:

$$\frac{\partial s(\boldsymbol{x},t)}{\partial t} = \frac{1}{2}b(t)\nabla_{\boldsymbol{x}} tr\frac{\partial s(\boldsymbol{x},t)}{\partial \boldsymbol{x}} + \frac{1}{2}b(t)\frac{\partial s(\boldsymbol{x},t)}{\partial \boldsymbol{x}}(\boldsymbol{x} + s(\boldsymbol{x},t)).$$

Given that the distribution at each time t_i and its associated score are intrinsically determined by the data distribution and the Forward Diffusion Process (FDP), we can expedite training by concurrently learning the scores $\nabla_{\boldsymbol{x}_{t_i}} \log p(\boldsymbol{x}_{t_i}, t_i)$ through each $s_{\boldsymbol{\theta}_i}(\boldsymbol{x}_{t_i})$ at all times t_i. If our models proficiently approximate these scores, the transformation resulting from combining these models will be smooth.

This instigates partitioning the diffusion process, specifically, the interval $t \in [0, 1]$ is divided into $\bigcup_{i=0}^{N-1}[t_i, t_{i+1}]$. Thereby, it remains feasible to train a time-dependent neural network $s_{\boldsymbol{\theta_i}}(\boldsymbol{x}_t, t)$ to acquire scores $\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t, t)$ for time $t \in [t_i, t_{i+1}]$. Such an approach diminishes the complexity of tasks for each model, bolstering modeling capability, whilst preserving time continuity. Moreover, given the parallelizability of training, a solitary model is loaded in memory per device during training, and post-training it can be stored on a disk and recalled at testing. The training procedure of this framework is described in Algorithm 2. In this case, each score modeling network $s_{\boldsymbol{\theta_i}}(\boldsymbol{x}_t, t)$ corresponds to a CNF block in the continuous normalizing flow representation.

Algorithm 2
Time-varying Parallel Score Matching (TPSM)

Input: data $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$, batch-size p
Split $[0, 1]$ into $\cup_{i=0}^{N-1}[t_i, t_{i+1}]$
for $i = 0$ **to** $N - 1$, in parallel **do**
  Train model $\boldsymbol{s_{\theta_i}}$
  repeat
    uniformly sample τ_1, ..., τ_p from $[t_i, t_{i+1}]$
    sample $\boldsymbol{x}^0_{\pi(1)}$, ..., $\boldsymbol{x}^0_{\pi(p)}$ from $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$
    generate $\boldsymbol{x}^{\tau_1}_{\pi(1)}$, ..., $\boldsymbol{x}^{\tau_p}_{\pi(p)}$ using Eq. 3.13
    minimize $\Sigma_j ||\boldsymbol{\varepsilon}_{\boldsymbol{\theta}_i}(\boldsymbol{x}^{\tau_j}_{\pi(j)}(\boldsymbol{\varepsilon_j}), \tau_j) - (\boldsymbol{\varepsilon_j})||^2$
  until convergence
end for

Algorithm 3
Discrete Parallel Score Matching (DPSM)

Input: data $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$, batch-size p
Discretize $[0, 1]$ into $\{t_0 = 0, t_1, ..., t_N = 1\}$
for $i = 1$ **to** N, in parallel **do**
  Train model $\boldsymbol{s_{\theta_i}}$
  repeat
    uniformly sample t_i from $\{t_0, ..., t_N\}$
    sample $\boldsymbol{x}^0_{\pi(1)}$, ..., $\boldsymbol{x}^0_{\pi(p)}$ from $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$
    generate $\boldsymbol{x}^{t_i}_{\pi(1)}$, ..., $\boldsymbol{x}^{t_i}_{\pi(p)}$ using Eq. 3.13
    minimize $\Sigma_j ||\boldsymbol{\varepsilon}_{\boldsymbol{\theta}_i}(\boldsymbol{x}^{t_i}_{\pi(j)}(\boldsymbol{\varepsilon_j})) - (\boldsymbol{\varepsilon_j})||^2$
  until convergence
end for

Elevating this approach to its apex, the duration of each interval approaches 0, thus we utilize a single network to model the score per time-point. This framework corresponds to an invertible residual flow with infinitesimal scaling, as invertibility and differentiability

are guaranteed by the fact that the learned transformation approximates the diffusion process. A description of this training procedure is given in Algorithm 3.

A comparison of the properties of Parallel Score Matching (PSM) with other generative diffeomorphism-based frameworks can be found in Table 3.1. Details about the generation procedure and likelihood estimation in the PSM framework can be found in Appendix B.1.

3.4 Experiments

We contrast the density estimation performance of DPMs trained via parallel score matching and those trained through the standard approach (SA-DPM). We conduct experiments on 2D data of toy distributions, alongside standard benchmark datasets including CIFAR-10, CelebA, and ImageNet.

For the 2D toy distribution data scenario, we solely compare the TPSM method against the baseline, namely, SA-DPM. The batch size across all experiments is 512, and we employ a learning rate of 10^{-3}. Each network was trained on a single CPU of equivalent performance and shares the same architecture in both approaches. This network is an MLP comprised of three hidden layers, wherein the non-linearity is provided by the ELU activation function.

Concerning standard image benchmark datasets, within the TPSM approach, we adopt the torch implementation [Wang, 2020] of the network originally proposed in [Ho, 2020] which, for the purpose of equitable comparison, is also utilized in SA-DPM. Similarly, within the DPSM approach, we opt for the even simpler basic U-Net, which originally presented the U-Net architecture in [Ronneberger, 2015]. For all models, we implement a batch size of 32 on ImageNet and CelebA, and a batch size of 128 for CIFAR-10. In DPSM we adopt a learning rate of 10^{-3}, while in TPSM and SA-DPM this is reduced to 2×10^{-4}. Each neural network across all three methods was trained on a single GPU with RTX-2080 level of performance. As in the 2D data experiments, we utilize the ELU activation function.

At no stage do we employ data parallelization, which constitutes an auxiliary parallelization strategy that complements the methodology presented in this chapter.

3.4.1 Toy 2D Datasets

To visually discern the distribution modeling capabilities between the baseline and the TPSM approach, we initially test both frameworks on 2D toy data. The two distributions

we examine are TY (refer to the upper row of Figure 3.2) and HG (lower row of Figure 2.5). The time-varying network deployed is identical in both methods and for both distributions. More specifically, it is an MLP with the subsequent structure: $4 \to 100 \to 150 \to 100 \to 2$. It merits mention that this model's input is 4-dimensional, as the model's time variability is enabled by appending the time component t to the 2D data input $\boldsymbol{x}(t)$. The output is 2-dimensional, as it predicts the score at point $\boldsymbol{x}(t)$ at time t, i.e., $[s_1(\boldsymbol{\theta}), s_2(\boldsymbol{\theta})] = \boldsymbol{s}_{\boldsymbol{\theta}} = \boldsymbol{s}_{\boldsymbol{\theta}}(x_1(t), x_2(t), t, t)$.

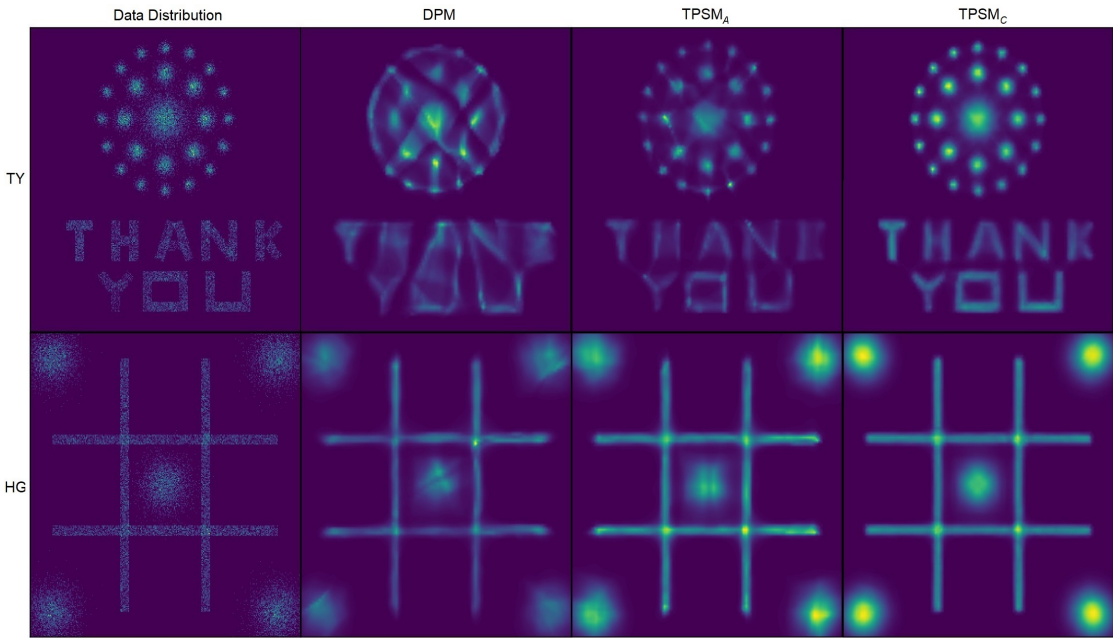

Fig. 3.2.: Probability density modeling capabilities of SA (second column), TPSM$_A$ (third column) and TPSM$_C$ (fourth column) on 2D data of toy distributions. The performance of TPSM$_B$ is shown in Appendix B.6.

As previously described, in SA-DPM the model attempts to learn all the scores of the evolution of the distribution dictated by the (DDPM) forward process:

$$d\boldsymbol{x}(t) = -\frac{1}{2}b(t)[\boldsymbol{x}(t) + \nabla_{\boldsymbol{x}(t)} \log p_t(\boldsymbol{x}(t))]dt; \; t \in [0, 1],$$

where we set $b(t) = 10t$.

Tab. 3.2.: A comparison of the results between SA-DPM and TPSM variants. The results are given in NLL (lower is better). Parallel training time is given in parentheses (blocks $\times$ hours per block).

Method:	SA-DPM	TPSM$_A$	TPSM$_B$	TPSM$_C$
TY	0.76 (1$\times$ **4h**)	0.69 (2$\times$ **1h**)	0.64 (4$\times$ **1h**)	0.58 (200$\times$ **1h**)
HG	1.11 (1$\times$ **1.3h**)	1.07 (2$\times$ **0.3h**)	1.04 (4$\times$ **0.3h**)	1.03 (200$\times$ **0.3h**)

In contrast, for the TPSM$_A$ approach, we train two models (networks), with the initial model spanning the time interval $[0, 0.1]$ and the subsequent one covering $[0.1, 1]$. In the case of TPSM$_B$, we partition the diffusion process into four subintervals with the following splits $\{0, 0.02, 0.1, 0.3, 1\}$. Conversely, in the extreme TPSM$_C$ approach, we utilize 200 such networks, where network i is trained to learn the scores corresponding to the evolution of the distribution dictated by the identical process, constrained to the time interval $t \in [\frac{i}{200}, \frac{i+1}{200}]$.

As evidenced in Figure 3.2, TPSM demonstrates a substantially enhanced performance in the evaluated tasks, when juxtaposed with the baseline (SA-DPM), notwithstanding the fact that the latter underwent a training duration four times lengthier. In the initial row, the intended target is the TY distribution, a notably intricate and arduous 2D distribution to model. It becomes apparent that the baseline exhibits difficulty in modeling such a distribution, as it fails to adequately delineate the distinct modalities constituting the probability density function. One could conjecture that a considerable enhancement of the model size would likely yield superior performance from the SA-DPM, albeit this would inevitably augment the already substantial training duration. Moreover, when considering more realistic scenarios involving high-dimensional data, the size of the network would inevitably be constrained due to memory-related limitations. Analogous distinctions can be discerned in the second row (HG distribution). In Table 3.2, we present the results of the test NLL for both models across both datasets, in addition to the difference in training time. The experiments demonstrated herein indicate that the TPSM approach is characterized by high flexibility of the vector field.

TPSM naturally retains the ability to generate samples from the learnt distribution and perform likelihood estimation by employing Equations (B.1) and (B.2) (Appendix B.1). In addition, any ODE solver can be used, including adaptive ones, identically as in the case of CNFs. In our experiments we use the Runge-Kutta-4 (RK4) method and we precisely calculate the divergence during validation, as there is no necessity to utilize the Hutchinson's trace estimator [Hutchinson, 1990; Grathwohl, 2019] due to the data's low dimensionality.

3.4.2 Image Datasets

In this section, we showcase the enhanced performance attained by employing PSM methods on image data. We parameterize the SA-DPM model using the time-varying U-Net architecture introduced in [Ho, 2020] and set the number of channels to 64. The same network is utilized in each block for the TPSM approach.

The initial TPSM variant explored, designated as TPSM0, follows a similar approach as TPSM$_A$ in the synthetic 2D data case, wherein the diffusion interval $[0, 1]$ is partitioned

into two subintervals: $[0, 0.1]$ and $[0.1, 1]$. By training each network in each block for half the number of parameter updates compared to the SA-DPM network, the training time is reduced by 50% through parallelization. Moreover, improvements in likelihood estimation results are observed (Table 3.3), with no adverse effects on inference time or memory usage, as illustrated in Tables B.4, B.5, and B.6 in Appendix B.2. During the inference phase, 100 steps were executed within the interval $[0, 0.1]$ and 900 steps within $[0.1, 1]$. Moreover, we evaluate TPSM1, which consists of 10 blocks, and to investigate

Tab. 3.3.: The results of TPSM0, where the two time subintervals have unequal length. In parentheses we give the number of GPUs (blocks) $\times$ the time of training per block.

Method:	CIFAR-10	CelebA 64×64	ImageNet 64×64
SA-DPM	3.13 (1× **72h**)	2.06 (1× **72h**)	3.62 (1× **180h**)
TPSM0	3.12 (2× **36h**)	1.92(2× **36h**)	3.55 (2× **90h**)

the limiting behavior of TPSM, we introduce TPSM2, comprised of 100 blocks. For TPSM1, block i models the score associated with times $t \in [\frac{i}{10}, \frac{i+1}{10}]$, whereas in the case of TPSM2, this changes to $t \in [\frac{i}{100}, \frac{i+1}{100}]$. The results are given in Table 3.4 and demonstrate significant improvements in density estimation and training time through parallelization. We observe that TPSM0 outperforms TPSM1, attributable to the decision to utilize the DDPM setting, where the majority of the local score evolution transpires when t approaches 0, [Nichol, 2021b]. This insinuates that the results of TPSM1 and TPSM2 could experience significant enhancements if more recent and efficient settings were employed, such as Flow-Matching, [Lipman, 2023], where the score's evolution is distributed more uniformly over time. Such is the case for 2D toy data as shown in Appendix B.5.

As in the case of 2D data, we set $b(t) = 10t$ in all cases. The RK4 solver (1k steps) was used during density estimation in SA-DPM, and in TPSM. If the Euler method is used, a higher number of integration steps is suggested ($\geq$ 5k), as otherwise sub-optimal results can be obtained, which overestimate the performance of the model. Generated samples from such models can be found in Appendix B.3. In Appendix B.2, we provide results in the case that SA-DPM and TPSM2 are trained for the same number of parameter updates.

Tab. 3.4.: Results comparing the performance of SA-DPM and the parallel score-matching approaches. We test the models on CIFAR-10, CelebA, and ImageNet (64x64). The results are given in bits/dim (lower is better), and the training time is given in parentheses.

Method:	CIFAR-10	CelebA 64×64	ImageNet 64×64
SA-DPM	3.13 (1× **72h**)	2.06 (1× **72h**)	3.62 (1× **180h**)
TPSM1	3.11 (10× **9h**)	2.07 (10× **9h**)	3.60 (10× **22h**)
TPSM2	2.93 (100× **4.5h**)	1.90 (100× **4.5h**)	3.55 (100× **14h**)
DPSM	2.93 (1000× **1.5h**)	1.94 (1000× **2.5h**)	3.59 (1000× **7h**)

Fig. 3.3.: Random non cherry-picked CIFAR-10 and CelebA samples generated by the DPSM with 1000 training/generative steps.

The model implemented in the DPSM framework differs significantly, as it does not depend on time. In this instance, we elect to employ the initial U-Net as introduced and implemented in the pioneering U-Net paper [Ronneberger, 2015]. The channel architecture is configured as follows: $(3, a, a*2, a*4, a*8, a*16, a*8, a*4, a*2, a, 3)$.

When a is set to 64, this architecture aligns with the standard implementation. The discretization process incorporates 1000 steps, signifying the use of 1000 basic U-Nets, each per step. The parameters $b(t)$ are defined as previously, $b(t) = 10t$. The classical non time-varying U-Net exhibits approximately 3.5 times faster training time per parameter update compared to the time-varying U-Net. Coupled with fewer parameter updates, this accelerates DPSM training, making it up to 50 times quicker than that of SA-DPM. Additionally, the U-Net's reduced complexity facilitates usage of batch sizes that are four times larger than in the time-varying scenario. However, DPSM implementation necessitates interpolation techniques for precise likelihood estimation. For our experiments involving a thousand networks, we simply interpret the ODE as a piece-wise constant function, specifically, in CNF block i, the vector field is defined to be constant, and is described by the optimized score approximator $s_i(\boldsymbol{\theta})$ at time t_i. Utilizing the Euler method, likelihood estimation is performed with 5k steps and generation with 1k steps, avoiding interpolation in the generation process. Figure 3.3 presents generated samples of CIFAR-10 and CelebA. Detailed information on the number of parameter updates for each model and dataset is available in Table B.2 in Appendix B.2. Although the main focus of this thesis is on density estimation performance quantified using rigorous scientific metrics like KL divergence (bits/dim), in Appendix B.2, we present results relating to the quality of generated samples, evaluated using the widely accepted 2-Wasserstein distance approximation (FID), [Seitzer, 2020]. Further experiments applying TPSM on time series for anomaly detection are given in Appendix B.7.

3.5 Limitations and Future Work

3.5.1 Number of Computing Units and Likelihood Estimation in DPSM

The DPSM framework fundamentally relies on parallel computing, which necessitates a substantial number of computing units. In fact, the true advantages of parallelization within DPSM can only be fully exploited when one has access to computing clusters. Conversely, the TPSM framework can be effectively operated even by those with access to a single GPU, provided that the number of blocks is kept low. As demonstrated earlier, training two CNF blocks sequentially (as in the TPSM0 scenario) requires the same duration as training a single block within the SA-DPM framework. However, this approach results in enhanced performance, as evidenced in Table 3.3.

Though the training process can be performed in parallel, the generation and likelihood estimation processes are inherently sequential. In the context of DPSM, it is necessary to interpolate between successive models to carry out accurate likelihood estimation. The generation process, on the other hand, remains unaffected.

3.5.2 Tailored TPSM Models

In this study, due to computational simplifications, we chose to employ smaller, older models across all three frameworks (SA-DPM, TPSM, DPSM). Our main objective was to demonstrate that the latter two methods outperform the first one. In all instances, for fair comparison, we used models in the individual blocks (steps) in the TPSM (DPSM) approach that were equal to or smaller than the ones used in SA-DPM. Looking ahead, it would be valuable to remove these restrictions and attempt to develop custom models for the TPSM/DPSM frameworks. This would allow us to assess how much these strategies can enhance the state of the art.

3.6 Conclusion

We have presented Parallel Score Matching strategies for training diffusion probabilistic models. We exploited the inherent properties of diffusion models which enable modeling of each score separately. We showed that learning different groups of scores in parallel via independent neural networks is effective and allows great improvements of training time, while enabling better model performance.

Part II

Tail Shape Estimation

4 On Tail Decay Rate Estimation of Loss Function Distributions

Contents

Tails constitute an important aspect of probability density functions, signifying infrequent yet substantial outcomes. Extreme Value Theory (EVT) represents a recognized domain tasked with the modeling of distribution tails. A principal parameter ascertained by traditional EVT methods such as Peaks-Over-Threshold is the tail-shape parameter, the determinant of the thickness of a distribution's tail. An intuitive query in this scenario is whether we can exploit supplementary information, provided by dependent random variables, to enhance the estimation of our variable of interest's tail shape. This question assumes notable relevance in specific contexts, such as those involving the loss-function distributions.

The study of loss-function distributions is critical to characterize a model's behaviour on a given machine-learning problem. While model quality is commonly measured by the

average loss assessed on a testing set, this quantity does not ascertain the existence of the mean of the loss distribution. Conversely, the existence of a distribution's statistical moments can be verified by examining the thickness of its tails.

Cross-validation schemes determine a family of testing loss distributions conditioned on the training sets. By marginalizing across training sets, we can recover the overall (marginal) loss distribution, whose tail-shape we aim to estimate. Small sample-sizes diminish the reliability and efficiency of classical tail-estimation methods like Peaks-Over-Threshold, and we demonstrate that this effect is notably significant when estimating tails of marginal distributions composed of conditional distributions with substantial tail-location variability. We mitigate this problem by utilizing a result we prove: under certain conditions, the marginal-distribution's tail-shape parameter is the maximum tail-shape parameter across the conditional distributions underlying the marginal. We label the resulting approach as 'cross-tail estimation (CTE)'.

We test CTE in a series of experiments on simulated and real data[1], showing the improved robustness and quality of tail estimation as compared to classical approaches.

4.1 Introduction

Loss function distributions form critical subjects of analysis, serving as barometers for machine learning model performance. In the context of a particular model and associated machine learning task, the true distribution of the loss function is typically elusive; we predominantly have access to a finite sample set, born from diverse choices of training and testing sets. To facilitate performance comparisons across different models based on the underlying loss function distributions, a spectrum of methodologies has been established. Traditional strategies derive from information criteria such as the Akaike Information Criterion (AIC) [Akaike, 1973; Akaike, 1974], an asymptotic approximation of the Kullback-Leibler divergence between the true data distribution and the fitting candidate, and its corrected version (AICc) [Sugiura, 1978; Hurvich, 1989], in addition to the Bayesian Information Criterion (BIC) [Schwarz, 1978]. The application of these information criteria, especially the AIC, is often constrained by the multiple inherent approximations and assumptions [Burnham, 2007], making them less feasible in certain scenarios. However, it warrants mention that more recent penalized criteria have considerably expanded their suitability for realistic setups [Birge, 1995; Arlot, 2009]. Simultaneously, other methodologies, termed splitting/resampling methods, have been devised, wherein a subset of the data is deployed to assess the performance of the trained model. This group of methodologies is expansive, predicated on a diverse range

[1]The code is available at https://github.com/ehaxholli/CTE

of partitioning and evaluation strategies addressing data heterogeneity and imbalance [Neyman, 1934; Cochran, 2007].

In the domain of cross-validation strategies, the common metric employed for gauging model performance is the sample mean of the loss function distribution. This practice, though invariably providing a finite numerical value, does not assure the existence of the first statistical moment or those of higher order. Moreover, this metric, in spite of its prevalence, should not necessarily be construed as a sole indicator of the model's performance, as it does not necessarily quantify its robustness to the underlying data distribution and model architecture. While it is true that aforementioned methods allow to rank models according to their relative performance on a given dataset, these scores still have limited value in quantifying the overall stability of a model. From a theoretical perspective, there is a connection between the uppermost existing moment of a distribution and the thickness of its tail. This underscores the significance of examining the behavioural traits and decay rate of the tails of loss function distributions.

In order to proceed, we first must be able to model the tails of distributions and to quantify their "thickness". Extreme Value Theory (EVT) is an established field concerned with modelling the tails of distributions. One of the fundamental results in EVT is the Pickands–Balkema–De Haan Theorem, which states that the tails of a large class of distributions can be approximated with generalized Pareto ones [Pickands, 1975; Haan, 2007]. In practice, the shape and scale parameter of the generalized Pareto are approximated from a finite sample, while its location parameter is always zero. It is the shape parameter which quantifies tail thickness, with larger values corresponding to heavier tails. The resulting estimation method is called Peaks-Over-Threshold (POT).

In the context of distributions of loss functions, for each training set, there is a corresponding conditional loss function distribution over points in the sample space. The actual total loss function distribution, the entity of our interest, is the weighted sum (integral) of all such conditional distributions, that is, it is the distribution created after marginalizing across the space of training datasets. In practice, we have a finite number of conditional distributions, as we have a finite number of training sets. Furthermore, for each of these conditional distributions, we only possess an approximation of them, derived from the samples in the testing set. The empirical approximation of the total loss function distribution therefore consists of the union of the sample sets of conditional distributions. Within this setting, the estimation of the tail shape of the total loss function distribution could be ideally carried out by applying POT on this union of samples.

In theory, as we show in this chapter, the role of the thickest conditional tails in determining the decay rate of the marginal is preserved, since the marginal and conditional distributions are defined everywhere, which allows the assessment of tails at extreme locations. Unfortunately, in practice, the finiteness of the sampling affects the estimation

of the tail of the marginal distribution, as the tails may be poorly or not even represented across different conditional distributions. To be more specific, during marginalization, samples from the tails of heavy tailed distributions can be overshadowed by the samples from the non-tail part of individual thin tailed ones. This suggests that modelling the tails of a marginal distribution by the usual application of POT can give inaccurate results in practice.

In this chapter, we develop a general method to mitigate the issue of estimating the tails of marginal distributions, when there exists a large variability between locations of the individual conditional distributions underlying the marginal. To this end, we demonstrate that under some regularity conditions, the shape parameter of the marginal distribution is precisely the maximum tail shape parameter of the family of conditional distributions. We refer to the method constructed from this result as *cross tail estimation*, due to similarities that it shares with Monte Carlo cross validation. The proposed solution enables a reduction in the sample size requirements, in the experiments we conducted. In the context of model comparison, our theory establishes that, under some assumptions, we can estimate the shape of the total loss distribution, by simply investigating the models prediction, without the need for target data. Furthermore, we show evidence of polynomial decay of tails of distributions of model predictions, and empirically demonstrate a relationship between the thickness of such tails and overfitting. An additional benefit of using the approach proposed here instead of the standard POT, is the reduced computational time in the case that the marginal is estimated from many conditional distributions.

The following is a summary of the structure of the chapter: In section 4.2 we recall some of the main concepts and results from Extreme Value Theory. In section 4.3, we state and generalize the main problem, which we tackle in section 4.4, by building our theory. We conclude section 4.4, by proving three statements which are useful for the experimental part, and by highlighting the relation between the tail of a distribution and its moments. In the final section, we show experimentally that our method can improve estimation in practice, as compared to the standard use of POT.

4.2 Related Work and Background

This section initially provides a succinct overview of Monte Carlo cross validation, given its conceptual similarities with the proposed method, “cross tail estimation". The subsequent subsection outlines standard results and definitions from extreme value analysis, forming the foundational bedrock for the proofs presented in section 4.4.

4.2.1 Monte Carlo Cross Validation

Let $D = \{(x_1, y_1), ...(x_n, y_n)\}$, be a set of data samples drawn form the same distribution. During each iteration i we sample k samples $D_i = \{(x_{\pi(1)}, y_{\pi(1)}), ..., (x_{\pi(k)}, y_{\pi(k)})\}$ without replacement from the original dataset D, and consider it as the training set for that iteration. The set $D \setminus D_i$ is then used as the testing set. The quantity of interest during iteration i is the sample mean of the loss of the model trained on D_i, namely $\hat{f}_{D_i}$, over the points of the testing set:

$$\tilde{M}_i^L := \frac{1}{|D \setminus D_i|} \sum_{j \in D \setminus D_i} L(\hat{f}_{D_i}(x_j), y_j), \tag{4.1}$$

for a given loss function L.
We evaluate the total performance of the model, based on its average performance over different choices of the training/testing sets, that is, the true evaluation metric is:

$$\tilde{M}^L := \frac{1}{m} \sum_{i=1}^{m} \tilde{M}_i^L = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{|D \setminus D_i|} \sum_{j \in D \setminus D_i} L(\hat{f}_{D_i}(x_j), y_j), \tag{4.2}$$

where m is the number of iterations (data partitions).
A detailed discussion on cross validation, elucidating its similarities with our proposed method for tail estimation in marginal loss function distributions, namely 'cross tail estimation', is presented in subsection 4.3.2.

4.2.2 Extreme Value Theory

Extreme value theory (EVT) or extreme value analysis (EVA) is a branch of statistics dealing with the extreme deviations from the median of probability distributions. Extreme value theory is closely related to failure analysis and dates back to 1923, when Richard von Mises discovered that the Gumbell distribution is the limiting distribution of the maximum of an iid sequence, sampled from a Gaussian distribution. In 1928, Ronald A. Fisher and Leonard H. C. Tippett in [Fisher, 1928], characterized the only three possible non-degenerate limiting distributions of the maximum in the general case: Frechet, Gumbel and Weibull. In 1943, Boris V. Gnedenko, gave a rigorous proof of this fact in [Gnedenko, 1943]. This result is known Fisher–Tippett–Gnedenko theorem, and forms the foundation of EVT. The three aforementioned limiting distributions of the maximum can be written in compact form and they are known as the class of extreme value distributions:

Definition 4.2.1. *The Generalized Extreme Value Distribution is defined as follows:*

$$G_{\xi,a,b}(x) = e^{-(1+\xi(ax+b))^{-\frac{1}{\xi}}}, \quad 1+\xi(ax+b) > 0, \tag{4.3}$$

where $b \in \mathbb{R}$, $\xi \in \mathbb{R} \setminus \{0\}$ and $a > 0$. For $\xi = 0$, we define the generalized Extreme Value Distribution as the limit when $\xi \to 0$, that is

$$G_{0,a,b}(x) = e^{-e^{-ax-b}}. \tag{4.4}$$

Theorem 4.2.2 (Fisher–Tippett–Gnedenko)**.** *: Let X be a real random variable with distribution F_X. Denote by $\{X_1, X_2, ..., X_n\}$ a set of iid samples from the distribution F_X, and define $M_n = \max\{X_1, ..., X_n\}$. If there exist two sequences $\{c_i > 0\}_{i\in\mathbb{N}}$ and $\{d_i \in \mathbb{R}\}_{i\in\mathbb{N}}$, such that*

$$c_n^{-1}(M_n - d_n) \xrightarrow{d} F \text{ as } n \to \infty, \tag{4.5}$$

for some non-degenerate distribution F, then we must have $F(x) = G_{\xi,a,b}(x)$, for some $b, \xi \in \mathbb{R}, a > 0$.

If X is a random variable as in Theorem 4.2.2, such that $F(x) = G_{\xi,a,b}(x)$, we say that F_X is in the Maximum Domain of Attraction of $G_{\xi,a,b}(x)$, and we write $F_X \in MDA(\xi)$. Depending on whether $\xi > 0$, $\xi = 0$, $\xi < 0$, we say that F_X is in the MDA of a Frechet, Gumbell, or Weibull distribution respectively.

Definition 4.2.3. *A Generalized Pareto distribution (GPD) with location parameter zero is defined as below:*

$$G_{\xi,\sigma}(w) = \begin{cases} 1 - (1 + \xi\frac{w}{\sigma}))^{-\frac{1}{\xi}} & \text{for } \xi \neq 0 \\ 1 - e^{-\frac{w}{\sigma}} & \text{for } \xi = 0 \end{cases}, \tag{4.6}$$

where $w > 0$ when $\xi > 0$ and $0 < w < -\frac{\sigma}{\xi}$ for $\xi < 0$. The shape parameter is denoted by ξ, while the scale parameter by $\sigma > 0$.

[Pickands, 1975], and [Balkema, 1974] proved that the limiting distribution of samples larger than a threshold is a Generalized Pareto distribution, whose location parameter is zero.

Theorem 4.2.4 (Pickands–Balkema–De Haan)**.** *: Let X be a random variable with distribution F_X and $x_F \leq \infty$ such that $\forall x > x_F$, $\bar{F}_X(x) = 0$. Then $F_X \in MDA(\xi) \iff \exists g : (0,\infty) \to (0,\infty)$ such that*

$$\lim_{u \to x_F} \sup_{y \in [0, x_F - u]} |\bar{F}_u^X(y) - \bar{G}_{\xi,g(u)}(y)| = 0, \tag{4.7}$$

where $\bar{F}_u^X(y) = \frac{1-F_X(y+u)}{1-F_X(u)}$.

This result forms the basis of the well-known Peak-Over-Threshold (POT) method which is used in practice to model the tails of distributions. The shape parameter can be estimated via different estimators such as the Pickands Estimator or the Deckers-Einmahl-de Haan Estimator (DEdH), [Dekkers, 1989a].

Definition 4.2.5. *Let* X_1, X_2,...,X_n *be iid samples from the distribution* F_X. *If we denote with* $X_{1,n}$, $X_{2,n}$,..., $X_{n,n}$ *the samples sorted in descending order, then the Pickands estimator is defined as follows:*

$$\hat{\xi}_{k,n}^{(P)} = \frac{1}{\ln 2} \ln \frac{X_{k,n} - X_{2k,n}}{X_{2k,n} - X_{4k,n}}. \tag{4.8}$$

Definition 4.2.6. *Let* X_1, X_2,...,X_n *be iid samples from the distribution* F_X. *If we denote with* $X_{1,n}$, $X_{2,n}$,..., $X_{n,n}$ *the samples sorted in descending order, then the DEdH estimator is defined as follows:*

$$\hat{\xi}_{k,n}^{(H)} = 1 + H_{k,n}^{(1)} + \frac{1}{2}\left(\frac{(H_{k,n}^{(1)})^2}{H_{k,n}^{(2)}} - 1\right)^{-1}, \tag{4.9}$$

where

$$H_{k,n}^{(1)} = \frac{1}{k}\sum_{j=1}^{k}(\ln X_{j,n} - \ln X_{k+1,n}) \tag{4.10}$$

and

$$H_{k,n}^{(2)} = \frac{1}{k}\sum_{j=1}^{k}(\ln X_{j,n} - \ln X_{k+1,n})^2. \tag{4.11}$$

An important result which we are going to use frequently in our proofs is Theorem 4.2.10, which can be found in [Embrechts, 2013; Haan, 2007], and gives the connection between the maximum domain of attraction and slowly varying functions.

Definition 4.2.7. *A positive measurable function L is called slowly varying if it is defined in some neighborhood of infinity and if:*

$$\lim_{x\to\infty} \frac{L(ax)}{L(x)} = 1, \textit{for all } a > 0. \tag{4.12}$$

Theorem 4.2.8 (Representation Theorem, see [Galambos, 1973])**.** *: A positive measurable function L on* $[x_0, \infty]$ *is slowly varying if and only if it can be written in the form:*

$$L(x) = e^{c(x)} e^{\int_{x_0}^{x} \frac{u(t)}{t} dt}, \tag{4.13}$$

where $c(t)$ and $u(t)$, are measurable bounded functions such that $\lim_{x\to\infty} c(x) = c_0 \in (0,\infty)$ and $u(t) \to 0$ as $t \to \infty$.

Proposition 4.2.9. *[Mikosch, 1999] If L is slowly varying then for every $\epsilon > 0$:*

$$\lim_{x\to\infty} x^{-\epsilon} L(x) = 0. \tag{4.14}$$

Proof. We give a proof in Appendix C.1 for the sake of completeness. □

Theorem 4.2.10. *: If $X \in MDA(\xi)$ and x_F is such that $\forall x > x_F,\ \bar{F}_X(x) = 0$ then:*

- $\xi > 0 \iff \bar{F}_X(x) = x^{-\frac{1}{\xi}} L(x)$*, where L is slowly varying,*
- $\xi < 0 \iff \bar{F}_X(x_F - \frac{1}{x}) = x^{\frac{1}{\xi}} L(x)$*, where L is slowly varying,*
- $\xi = 0 \iff \bar{F}_X(x) = c(x) e^{-\int_w^x \frac{1}{a(t)} dt},\ w < x < x_F \leq \infty$*, where c is a measurable function satisfying $c(x) \to c > 0$ as $x \uparrow x_F$, and $a(x)$ is a positive, absolutely continuous function (with respect to Lebesgue measure) with density $a'(x)$ having $\lim_{x\uparrow x_F} a'(x) = 0$. If $x_F < \infty$ then $\lim_{x\uparrow x_F} a(x) = 0$ as well.*

4.3 Setup and Problem Statement

In the initial subsection, we establish a formal framework to address the problem of tail modelling for total loss distributions, and elucidate how unsatisfactory results can arise from a naive application of the Peaks-Over-Threshold (POT) method. Subsequently, in the second subsection, we present Cross-Tail-Estimation (CTE), a novel methodology that addresses these shortcomings. A salient feature of this section is the illustration of the analogy between CTE and Cross-Validation, providing an intuitive understanding of CTE. In the concluding subsection, we lay the groundwork for the upcoming section 4.4. In this forthcoming section, we provide the theoretical justification, in the form of Theorem 4.4.11, for the application of our introduced method, CTE.

4.3.1 Problem Statement

We assume that each data sample $(\boldsymbol{X}, \boldsymbol{Y})$ comes from distribution $\mathcal{D}$ and that the sampling is independent. We use the symbol $\boldsymbol{X}$ to denote the features and the symbol $\boldsymbol{Y}$ to denote the labels. The training set will be defined as a random vector comprised of iid random vectors $(\boldsymbol{X}, \boldsymbol{Y})$ sampled from $\mathcal{D}$. More precisely, after fixing a natural number

k, we define a training set as $\boldsymbol{V} = [(\boldsymbol{X}, \boldsymbol{Y})_1, (\boldsymbol{X}, \boldsymbol{Y})_2, ..., (\boldsymbol{X}, \boldsymbol{Y})_k]$, where each $(\boldsymbol{X}, \boldsymbol{Y})_i$ has distribution $\mathcal{D}$. On the other hand, a test point $\boldsymbol{U}$ naturally is defined as a sample from $\mathcal{D}$, i.e., $\boldsymbol{U} = (\boldsymbol{X}, \boldsymbol{Y})$. In practice, the realisation of $\boldsymbol{U}$ should not be an entry in $\boldsymbol{V}$. A model which is trained on $\boldsymbol{V}$ to predict $\boldsymbol{Y}$ from $\boldsymbol{X}$ is denoted as $\hat{\boldsymbol{h}}_{\boldsymbol{V}}(\boldsymbol{X})$. The prediction error on the testing datum $\boldsymbol{U}$ of a model trained on $\boldsymbol{V}$ is denoted as $W_{\boldsymbol{V}}(\boldsymbol{U})$. For the remainder of the chapter we assume that $W_{\boldsymbol{V}}(\boldsymbol{U}) > 0$ and notice that the probability density function of $W_{\boldsymbol{V}}(\boldsymbol{U})$ is

$$f_W(w) = \int f_{W,\boldsymbol{V}}(w, \boldsymbol{v})d\boldsymbol{v} = \int f_{\boldsymbol{V}}(\boldsymbol{v})f(w|\boldsymbol{V} = \boldsymbol{v})d\boldsymbol{v} = \int f_{\boldsymbol{V}}(\boldsymbol{v})f_{\boldsymbol{v}}(w)d\boldsymbol{v}, \tag{4.15}$$

therefore the distribution function of $W_{\boldsymbol{V}}(\boldsymbol{U})$ is:

$$F_W(w) = \int f_{\boldsymbol{V}}(\boldsymbol{v})F_{\boldsymbol{v}}(w)d\boldsymbol{v}. \tag{4.16}$$

$F_{\boldsymbol{v}}(w)$ is the distribution of the prediction error (loss) of the model trained on training set $\boldsymbol{v}$, while $F_W(w)$ is the unconditional distribution of the loss.

Standard methods such as the Peaks-Over-Threshold (POT) approach, when employed directly for estimating the tails of general marginal distributions like $F_W(w)$ in Equation (4.16), may yield unsatisfactory outcomes.

To provide insight into the problem, let's simplify the scenario for a moment, by assuming that some random vector $\boldsymbol{V}$ can be either $\boldsymbol{v}_1$ or $\boldsymbol{v}_2$, each with an equal likelihood. In the situation where $\boldsymbol{V} = \boldsymbol{v}_1$, assume $F_{\boldsymbol{v}_1}(w)$ corresponds to a thick-tailed distribution, wherein even the first moment does not exist. Conversely, if $\boldsymbol{V} = \boldsymbol{v}_2$, suppose $F_{\boldsymbol{v}_2}(w)$ takes the form of a Gaussian distribution, characterized by a large mean. Under these conditions, Equation (4.16) simplifies to $F_W(w) = \frac{1}{2}F_{\boldsymbol{v}_1}(w) + \frac{1}{2}F_{\boldsymbol{v}_2}(w)$. It is known that the tail shape parameter of $F_W(w) = \sum_{i=1}^{n} p(\boldsymbol{v}_i)F_{\boldsymbol{v}_i}(w)$ is determined by the conditional distribution $F_{\boldsymbol{v}_i}(w)$ with the thickest tail. In our case, n above is 2, and the tail of $F_W(w)$ is defined by the fat tail of $F_{\boldsymbol{v}_1}(w)$. Suppose we proceed with the standard POT approach, that is, we integrate out the random variable $\boldsymbol{V}$, and subsequently estimate the shape parameter of the tail of $F_W(w)$. In practical scenarios, this translates to merging the samples from both conditional distributions into a singular array. Given the finite nature of sample sizes in such cases, it's conceivable that none of the samples of W from the thick-tailed distributions surpass those from the Gaussian distribution, owing to the discrepancies in their locations. As a consequence, the sample tail of the marginal (mixture) distribution takes its shape from the sample tail of the Gaussian $F_{\boldsymbol{v}_2}(w)$, while in reality, the tail of $F_W(w)$ is dictated by the heavy tail of $F_{\boldsymbol{v}_1}(w)$. In the ideal scenario with limitless sampling, we would expect to determine the true tail shape. Yet, within the constraints of practical applications, it may be necessary to estimate the tail shape parameters of $F_{\boldsymbol{v}_1}(w)$ and $F_{\boldsymbol{v}_2}(w)$ individually.

A natural question that arises in the case that $\boldsymbol{V}$ has a continuous distribution as in Equation (4.16) is whether the tail of the marginal $F_W(w)$ is still determined by the largest tail of the conditional distributions $F_{\boldsymbol{v}}(w)$. As we will prove in section 4.4, under some regularity conditions, the answer is in the affirmative.

4.3.2 Cross Tail Estimation

We will denote with ξ_v the tail shape parameter of $F_{\boldsymbol{v}}(w)$ and with ξ the shape tail parameter of $F_W(w)$. Our goal in section 4.4 is to prove that under some regularity conditions if $\exists \boldsymbol{v}$, such that $\xi_{\boldsymbol{v}} > 0$, then $\xi = \max\{\xi_{\boldsymbol{v}} | \boldsymbol{v}\}$, and if $\forall \xi_{\boldsymbol{v}} \leq 0$, then we have $\xi \leq 0$. This motivates Algorithm 4 which we name "Naive Cross Tail Estimation" (NCTE).

Algorithm 4 Naive Cross Tail Estimation

Input: Data $D = [(\boldsymbol{x}, \boldsymbol{y})_1, (\boldsymbol{x}, \boldsymbol{y})_2, ..., (\boldsymbol{x}, \boldsymbol{y})_n]$; the Pickands or DEdH estimator
Define: $A = \{\}$
Fix the number of training sets (rounds): $m \in \mathbb{N}$
repeat
1. sample $(\boldsymbol{x}, \boldsymbol{y})_{\pi(1)}$, ..., $(\boldsymbol{x}, \boldsymbol{y})_{\pi(k)}$ from $(\boldsymbol{x}, \boldsymbol{y})_1, (\boldsymbol{x}, \boldsymbol{y})_2, ..., (\boldsymbol{x}, \boldsymbol{y})_n$
2. train model $\hat{\boldsymbol{h}}_{\boldsymbol{v}}$ on $\boldsymbol{v} = [(\boldsymbol{x}, \boldsymbol{y})_{\pi(1)}, ..., (\boldsymbol{x}, \boldsymbol{y})_{\pi(k)}]$
3. calculate the prediction errors $W_{\boldsymbol{v}}(\boldsymbol{U})$ of model $\hat{\boldsymbol{h}}_{\boldsymbol{v}}$ on the testing set $D \setminus \boldsymbol{v}$
4. group the calculated prediction errors in the set $E_{\boldsymbol{v}}(D)$
5. apply the Pickands or DEdH estimator on $E_{\boldsymbol{v}}(D)$ to estimate $\xi_{\boldsymbol{v}}$
6. add $\hat{\xi}_{\boldsymbol{v}}$ to A

until $|A| = m$
return $\max A$ if $\max A > 0$, else return 'non-positive'

Since for each $\boldsymbol{v}$, the estimated $\hat{\xi}_{\boldsymbol{v}}$ is prone to estimation errors, taking the maximum $\hat{\xi}_{\boldsymbol{v}}$ over all $\boldsymbol{v}$ tends to cause NCTE to overestimate the true ξ, especially when the number of conditional distributions $F_{\boldsymbol{v}}(w)$ is large. For this reason we propose Algorithm 5, named 'Cross Tail Estimation' (CTE), where we split the samples from $F_{\boldsymbol{v}}(w)$ into p sets in order to get p estimates of the tail shape parameter of $F_{\boldsymbol{v}}(w)$, that is $\{\hat{\xi}^1_{\boldsymbol{v}}, \hat{\xi}^2_{\boldsymbol{v}}, ..., \hat{\xi}^p_{\boldsymbol{v}}\}$. Our final estimation of $\xi_{\boldsymbol{v}}$ is the average of the p estimations, i.e., $\frac{1}{p}\sum_{i=0}^{p} \hat{\xi}^i_{\boldsymbol{v}}$. A more detailed justification for utilizing Algorithm 5 is given in Appendix C.5. We notice that Algorithm 5 is identical to Algorithm 4 when $p = 1$.

Remark: Estimating particular statistics of $F_W(w)$ through the statistics of $F_{\boldsymbol{v}}(w)$ as in in Algorithm 4 and 5 is a key component of Cross Validation. During Cross Validation, a training set $\boldsymbol{v}$ and a testing set $D \setminus \boldsymbol{v}$ are selected in each iteration, during which the following conditional expectation is then estimated:

$$\mathbb{E}[W_{\boldsymbol{V}}(\boldsymbol{U}) | \boldsymbol{V} = \boldsymbol{v}] = \int w f_{\boldsymbol{v}}(w) dw. \tag{4.17}$$

Algorithm 5 Cross Tail Estimation

Input: Data $D = [(\boldsymbol{x},\boldsymbol{y})_1, (\boldsymbol{x},\boldsymbol{y})_2, ..., (\boldsymbol{x},\boldsymbol{y})_n]$; the Pickands or DEdH estimator
Define: $A = \{\}$
Fix the number of training sets (rounds): $m \in \mathbb{N}$
repeat

1. sample $(\boldsymbol{x},\boldsymbol{y})_{\pi(1)}$, ..., $(\boldsymbol{x},\boldsymbol{y})_{\pi(k)}$ from $(\boldsymbol{x},\boldsymbol{y})_1, (\boldsymbol{x},\boldsymbol{y})_2, ..., (\boldsymbol{x},\boldsymbol{y})_n$
2. train model $\hat{\boldsymbol{h}}_{\boldsymbol{v}}$ on $\boldsymbol{v} = [(\boldsymbol{x},\boldsymbol{y})_{\pi(1)}, ..., (\boldsymbol{x},\boldsymbol{y})_{\pi(k)}]$
3. calculate the prediction errors $W_{\boldsymbol{v}}(\boldsymbol{U})$ of model $\hat{\boldsymbol{h}}_{\boldsymbol{v}}$ on the testing set $D \setminus \boldsymbol{v}$
4. group the calculated prediction errors in the set $E_{\boldsymbol{v}}(D)$
5. split $E_{\boldsymbol{v}}(D)$ into $\{E^1_{\boldsymbol{v}}(D), ..., E^p_{\boldsymbol{v}}(D)\}$
6. apply the Pickands or DEdH estimator on each $E^i_{\boldsymbol{v}}(D)$ to get an estimate $\hat{\xi}^i_{\boldsymbol{v}}$ of $\xi_{\boldsymbol{v}}$
7. average over p to get the final estimate $\hat{\xi}_{\boldsymbol{v}} = \frac{1}{p}\sum_{i=0}^{p} \hat{\xi}^i_{\boldsymbol{v}}$ of $\xi_{\boldsymbol{v}}$
8. add $\hat{\xi}_{\boldsymbol{v}}$ to A

until $|A| = m$
return $\max A$ if $\max A > 0$, else return 'non-positive'

The estimates of $\mathbb{E}[W_{\boldsymbol{V}}(\boldsymbol{U})|\boldsymbol{V}]$ received in each iteration are then averaged to get an estimation of the total expectation:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{U},\boldsymbol{V}}(W_{\boldsymbol{V}}(\boldsymbol{U})) &= \int w f(w) dw = \int f_{\boldsymbol{V}}(\boldsymbol{v}) \int w f_{\boldsymbol{v}}(w) dw d\boldsymbol{v} = \\ &= \int f_{\boldsymbol{V}}(\boldsymbol{v}) \mathbb{E}[W_{\boldsymbol{V}}(U)|\boldsymbol{V} = \boldsymbol{v}] d\boldsymbol{v} = \mathbb{E}[\mathbb{E}[W_{\boldsymbol{V}}(\boldsymbol{U})|\boldsymbol{V} = \boldsymbol{v}]].\end{aligned} \tag{4.18}$$

In the language of section 4.3.1, the mean of distribution $F_W(w)$ is the average of the means of the conditional distributions $F_{\boldsymbol{v}}(w)$.
This statement about sums stands parallel with our claim about extremes: the shape parameter of the tail of $F_W(w)$, if positive, is the maximum of the shape parameters of the tails of the conditional distributions $F_{\boldsymbol{v}}(w)$.

4.3.3 The General Problem

Generalizing the problem stated in section 4.3.1 requires considering a one dimensional random variable of interest X, dependent on other random variables $\{Z_1, Z_2, ..., Z_n\}$, such that the probability density function of X is

$$f_X(x) = \int f(z_1, ..., z_n, x) d_{z_1} \cdots d_{z_n} \tag{4.19}$$

$$= \int f(\boldsymbol{z}) f(x|\boldsymbol{z}) d\boldsymbol{z} = \int f(\boldsymbol{z}) f_{\boldsymbol{z}}(x) d\boldsymbol{z}. \tag{4.20}$$

Integrating with respect to x we get

$$F_X(x) = \int f(\boldsymbol{z}) F(x|\boldsymbol{z}) d\boldsymbol{z} = \int f(\boldsymbol{z}) F_{\boldsymbol{z}}(x) d\boldsymbol{z}. \tag{4.21}$$

In this case, with regards to the previous section, we notice that $\boldsymbol{Z} = \boldsymbol{V}$ is the training set on which we condition, while $X = W$ is the random variable of interest. In section 4.4, we give several results which relate the tails of $F_X(x)$ and $F(x|\boldsymbol{z})$, culminating with Theorem 4.4.11 which justifies the usage of the CTE algorithm, by providing limiting behaviour guarantees.

4.4 Theoretical Results

In this section, we build our theory of modelling the tails of marginal distributions, which culminates with Theorem 4.4.11. We conclude this section by proving three statements which are useful in the experimental section 4.5, and give the relation between the existence of the moments of a distribution and the thickness of its tails. Unless stated otherwise, the proofs of all the statements are given in Appendix C.1.

4.4.1 Tails of marginal distributions

For two given distributions, whose tails have positive shape parameters, we expect the one with larger tail parameter to decay slower. Indeed:

Lemma 4.4.1. *If* $F_1 \in MDA(\xi_1)$ *and* $F_2 \in MDA(\xi_2)$*, and if* $\xi_1 > \xi_2 > 0$*, then* $\lim_{x\to\infty} \frac{\bar{F}_2(x)}{\bar{F}_1(x)} = 0$.

In a similar fashion, regardless of the signs of the shape parameters, we expect the one with larger tail parameter to decay slower. In fact we have the following:

Lemma 4.4.2. *If* $F_1 \in MDA(\xi_1)$ *and* $F_2 \in MDA(\xi_2)$ *then:*

1. *If* $\xi_1 > 0$ *and* $\xi_2 = 0$ *then* $\lim_{x\to\infty} \frac{\bar{F}_2(x)}{\bar{F}_1(x)} = 0$.
2. *If* $\xi_1 = 0, x_{F_1} = \infty$ *and* $\xi_2 < 0$ *then* $\lim_{x\to\infty} \frac{\bar{F}_2(x)}{\bar{F}_1(x)} = 0$.
3. *If* $\xi_1 > 0$ *and* $\xi_2 < 0$ *then* $\lim_{x\to\infty} \frac{\bar{F}_2(x)}{\bar{F}_1(x)} = 0$.

Despite the fact that a linear combination of slowly varying functions is not necessarily slowly varying, the following statement holds true:

Lemma 4.4.3. *If for $i \in \{1, ..., n\}$ we let $L_i(x)$ be slowly varying functions, and $\{a_1, ..., a_n\}$ be a set of positive real numbers, then*

$$L(x) = \sum_{i=1}^{n} a_i L_i(x)$$

is slowly varying.

In the case of a mixture of a finite number of distributions the following known result holds:

Theorem 4.4.4. *Let $Z : \Omega \to A \subset \mathbb{R}^n$ be a random vector where $|A| < \infty$. At each point $\boldsymbol{z}_1, .., \boldsymbol{z}_n \in A$, we define a distribution $F_{\boldsymbol{z}_i}(x) \in MDA(\xi_i)$ and assume that $\xi_{\max} := \max(\xi_1 = \xi_{z_1}, ..., \xi_n = \xi_{z_n}) > 0$. If the set $\{p_1, ..., p_n\}$ is a set of convex combination parameters, that is $\sum_i p_i = 1$ and $p_i > 0$ then:*

$$F(x) = \sum_{i}^{n} p_i F_{\boldsymbol{z}_i}(x) \in MDA(\xi_{\max}). \tag{4.22}$$

If $\xi_{\max} \leq 0$ then if ξ_F exists we have $\xi_F \leq 0$.

Proof. While this result is well known, we give an alternative proof in Appendix C.1, using the Pickands-Balkema-De Haan Theorem. □

From now on, we assume that the functions $F_A(x) = \int_A f_{\boldsymbol{Z}}(\boldsymbol{z}) F_{\boldsymbol{z}}(x) d\boldsymbol{z}$ defined on any element A of the Borel $\sigma - algebra$ induced by the usual metric are in the MDA of some extreme value distribution. Furthermore, we assume that the pdf $f_{\boldsymbol{Z}}(\boldsymbol{z})$ is strictly positive everywhere in its domain.

Proposition 4.2.9 states that every slowly varying function is sub-polynomial. That is for any $\delta > 0$ and any slowly varying function $L(x)$, if we are given any $\gamma > 0$, then we can find $x(L, \delta, \gamma) > 0$, such that for all $x > x(L, \delta, \gamma)$, the inequality $x^{-\delta} L(x) < \gamma$ holds. However, since $x(L, \delta, \gamma)$ depends on the function L, assuming that we have a family of $\{L_{\boldsymbol{z}} | \boldsymbol{z} \in A\}$, where A is a measurable set, the set $\{x(L_{\boldsymbol{z}}, \delta, \gamma) | \boldsymbol{z} \in A\}$ can be unbounded, suggesting that the beginning of the tail of $\bar{F}_{\boldsymbol{z}}(x) = x^{-\frac{1}{\xi_{\boldsymbol{z}}}} L_{\boldsymbol{z}}(x)$ can be postponed indefinitely across the family $\{F_{\boldsymbol{z}} | \boldsymbol{z} \in A\}$. These concepts are formalized in the following:

Definition 4.4.5. *For a set A, the family of sub-polynomial functions $\{L_{\boldsymbol{z}}(x) | \boldsymbol{z} \in A\}$ is called γ-uniformly sub-polynomial if for any fixed $\delta > 0$, there exists a $\gamma(\delta)$ so that the set*

$\{x_0|\boldsymbol{z} \in A\}$ is bounded from above, where $x_0 = x_0(L_{\boldsymbol{z}}, \delta, \gamma)$ is the smallest value for which when $x > x_0$ we have $x^{-\delta} L_{\boldsymbol{z}}(x) < \gamma$.

Proposition 4.4.6. *Let $\boldsymbol{Z} : \Omega \to A \subset \mathbb{R}^n$ be a random vector where A is measurable and define a family of sub-polynomial functions $\{L_{\boldsymbol{z}}(x)|\boldsymbol{z} \in A\}$, which we assume is γ-uniformly sub-polynomial. Then for a probability density function $f_{\boldsymbol{Z}}(\boldsymbol{z})$ on A induced by $\boldsymbol{Z}$, the function $L(x) = \int_A f_{\boldsymbol{Z}}(\boldsymbol{z}) L_{\boldsymbol{z}}(x) d\boldsymbol{z}$ is sub-polynomial.*

In the following theorem, we assume that all conditional distributions have positive tail shape parameters, and we show that the marginal distribution cannot have a tail shape parameter larger (smaller) than the largest (smallest) tail shape parameter across conditional distributions. Furthermore, if the tail shape parameters vary continuously across the space of conditional distributions, then the tail shape parameter of the marginal is precisely the same as the maximal tail shape parameter of the conditional distributions.

Theorem 4.4.7. *Let $\boldsymbol{Z} : \Omega \to A \subset \mathbb{R}^n$ be a random vector where A is measurable. At each point $\boldsymbol{z} \in A$ define a distribution $F_{\boldsymbol{z}}(x) \in MDA(\xi_{\boldsymbol{z}})$, and suppose there exist $\xi_{lo},\ \xi_{up}$ such that $\forall \boldsymbol{z} \in A,\ 0 < \xi_{lo} \leq \xi_{\boldsymbol{z}} \leq \xi_{up}$. Furthermore, let $L_{\boldsymbol{z}}(x)$ be the slowly varying function corresponding to $F_{\boldsymbol{z}}(x)$. If the family $\{L_{\boldsymbol{z}}(x)|\boldsymbol{z} \in A\}$ is γ-uniformly sub-polynomial, then for $F(x) = \int_A f_{\boldsymbol{Z}}(\boldsymbol{z}) F_{\boldsymbol{z}}(x) d\boldsymbol{z}$ we have $\xi_{lo} \leq \xi_F \leq \xi_{up}$. Furthermore, if $\xi_{\boldsymbol{z}}$ is continuous in $\boldsymbol{z}$, then $\xi_F = \xi_{\max}$, where $\xi_{\max} := \sup\{\xi_{\boldsymbol{z}}|\boldsymbol{z} \in A\}$.*

Similarly to the case when $F_{\boldsymbol{z}}(x)$ are in the $MDA(\xi_{\boldsymbol{z}})$ for $\xi_{\boldsymbol{z}} > 0$, if we wish to extend the results above, regularity conditions are required for the $\xi_{\boldsymbol{z}} \leq 0$ case. We notice that if $F_z(x) \in MDA(\xi)$ for $\xi \leq 0$, then $\bar{F}_{\boldsymbol{z}}(x)$ itself is sub-polynomial, whether its support is bounded or not. This observation motivates the following:

Definition 4.4.8. *For a set A, define the family of distribution functions $\mathcal{F}_{\mathcal{A}} = \{F_{\boldsymbol{z}}(x)|\boldsymbol{z} \in A\}$, and define $A^+ = \{\boldsymbol{z}|\xi_{\boldsymbol{z}} > 0\},\ A^- = \{\boldsymbol{z}|\xi_{\boldsymbol{z}} \leq 0\}$. We say family $\mathcal{F}_{\mathcal{A}}$ has stable cross-tail variability if,*

- *$\{L_{\boldsymbol{z}}(x)|\boldsymbol{z} \in A^+\}$ is γ-uniformly sub-polynomial,*
- *$\{\bar{F}_{\boldsymbol{z}}(x)|\boldsymbol{z} \in A^-\}$ is γ-uniformly sub-polynomial.*

We notice that in the previous theorem, if for all $\boldsymbol{z}$ we have $0 < \xi_{\boldsymbol{z}} \leq \epsilon$, then $\xi_F \leq \epsilon$. If the corresponding family $\mathcal{F}_{\mathcal{A}} = \{F_{\boldsymbol{z}}(x)|\boldsymbol{z} \in A\}$ has stable cross-tail variability, this holds independently from the lower bound of $\{\xi_{\boldsymbol{z}}|\boldsymbol{z} \in A\}$. Indeed:

Lemma 4.4.9. *Let $\boldsymbol{Z} : \Omega \to A \subset \mathbb{R}^n$ be a random vector where A is measurable. At each point $\boldsymbol{z} \in A$ define a distribution $F_{\boldsymbol{z}}(x) \in MDA(\xi_{\boldsymbol{z}})$, and suppose that $\forall \boldsymbol{z} \in A,\ \xi_{\boldsymbol{z}} \leq \epsilon$. If the family $\{F_{\boldsymbol{z}}(x) | \boldsymbol{z} \in A\}$ has stable cross-tail variability, then for $F(x) = \int_A f_{\boldsymbol{Z}}(\boldsymbol{z}) F_{\boldsymbol{z}}(x) d\boldsymbol{z}$ we have $\xi_F \leq \epsilon$.*

Corollary 4.4.10. *Let $\boldsymbol{Z} : \Omega \to A \subset \mathbb{R}^n$ be a random vector where A is measurable. At each point $\boldsymbol{z} \in A$ define a distribution $F_{\boldsymbol{z}}(x) \in MDA(\xi_{\boldsymbol{z}})$, and suppose that $\forall \boldsymbol{z} \in A,\ \xi_{\boldsymbol{z}} \leq 0$. If the family $\{F_{\boldsymbol{z}}(x) | \boldsymbol{z} \in A\}$ has stable cross-tail variability, then for $F(x) = \int_A f_{\boldsymbol{Z}}(\boldsymbol{z}) F_{\boldsymbol{z}}(x) d\boldsymbol{z}$ we have $\xi_F \leq 0$.*

Proof. We notice that for any $\epsilon > 0$, we have $\xi_{\boldsymbol{z}} < \epsilon$ for all $\boldsymbol{z} \in A$. Hence, from the previous Lemma we conclude that $\xi_F \leq \epsilon, \forall \epsilon > 0$. □

Finally, we prove the generalization of Theorem 4.4.7 in the case that the tail shape parameters $\xi_{\boldsymbol{Z}}$ of the conditional distributions are real numbers:

Theorem 4.4.11. *Let $\boldsymbol{Z} : \Omega \to A \subset \mathbb{R}^n$ be a random vector where A is measurable. At each point $\boldsymbol{z} \in A$ define a distribution $F_{\boldsymbol{z}}(x) \in MDA(\xi_{\boldsymbol{z}})$, where $\xi_{\boldsymbol{z}}$ is continuous and $\xi_{\max} > 0$. If the family $\{F_{\boldsymbol{z}}(x) | \boldsymbol{z} \in A\}$ has stable cross-tail variability, then for $F(x) = \int_A f_{\boldsymbol{Z}}(\boldsymbol{z}) F_{\boldsymbol{z}}(x) d\boldsymbol{z}$ we have $\xi_F = \xi_{\max}$. In the case that $\xi_{\max} \leq 0$ then $\xi_F \leq 0$.*

Examples when the conditions of Theorem 4.4.11 hold, as well as when they are violated, can be found in Appendix C.3 and Appendix C.2, respectively.

4.4.2 Useful propositions for the experimental part

In this subsection, we prove three statements which are useful in the experimental section 4.5, and state the well-known relation between the existence of the moments of a distribution and the thickness of its tails.

Proposition 4.4.12. *Let F_X be the distribution of the random variable X. We define X_1 to be a random variable whose distribution is the normalized right tail of F_X, that is:*

$$F_{X_1}(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ \frac{F(x) - F(0)}{1 - F(0)} & \text{for } x > 0 \end{cases}. \tag{4.23}$$

Similarly we define X_2 whose distribution is the normalized left tail of F_X,

$$F_{X_2}(x) = \begin{cases} 0 & \text{for } x < 0 \\ \frac{F(0)-F(-x)}{F(0)} & \text{for } x \geq 0 \end{cases}. \tag{4.24}$$

If $F_{X_1} \in MDA(\xi_1)$, $F_{X_2} \in MDA(\xi_2)$, and $\max\{\xi_1, \xi_2\} > 0$, then:

$$\xi_{|X|} = \max\{\xi_1, \xi_2\}.$$

If $F_{X_1} \in MDA(\xi_1)$, $F_{X_2} \in MDA(\xi_2)$, and $\max\{\xi_1, \xi_2\} \leq 0$, then:

$$\xi_{|X|} \leq 0.$$

Proof. Since

$$\begin{aligned} F_{|X|}(x) = \mathbb{P}(|X| < x) &= \mathbb{P}(X < x | X > 0)\mathbb{P}(X > 0) + \mathbb{P}(-X < x | X \leq 0)\mathbb{P}(X \leq 0) \\ &= p_1 F_{X_1}(x) + p_2 F_{X_2}(x), \end{aligned} \tag{4.25}$$

Theorem 4.4.4 gives the desired conclusion. □

Proposition 4.4.13. *Let X be a random variable such that $X \in MDA(\xi_X > 0)$. If we define Y to be equal to X^α, for some $\alpha \in \mathbb{R}^+$, then $Y \in MDA(\xi_Y)$ where $\xi_Y = \alpha\xi_X$. If $\xi_X \leq 0$ then $\xi_Y \leq 0$.*

It is important to notice that we can estimate the shape of the tail of $W_{\boldsymbol{V}}(\boldsymbol{U})$ by also conditioning on the test label $\boldsymbol{y}$:

$$f_W(w) = \int f_{W,\boldsymbol{Y}}(w, \boldsymbol{y})d\boldsymbol{y} = \int f_{\boldsymbol{Y}}(\boldsymbol{y}) f(w|\boldsymbol{Y} = \boldsymbol{y})d\boldsymbol{y} = \int f_{\boldsymbol{Y}}(\boldsymbol{y}) f_{\boldsymbol{y}}(w)d\boldsymbol{y} \tag{4.26}$$

$$F_W(w) = \int f_{\boldsymbol{Y}}(\boldsymbol{y}) F_{\boldsymbol{y}}(w)d\boldsymbol{y}. \tag{4.27}$$

We use this fact to prove the following:

Proposition 4.4.14. *Let the loss function be defined as $W_{\boldsymbol{V}}(\boldsymbol{U}) = |Y - \hat{f}_{\boldsymbol{V}}(\boldsymbol{X})|^p$ for some $p \in \mathbb{R}^+$, and let $F_y(t)$ be the distribution of $\hat{f}_{\boldsymbol{V}}(\boldsymbol{X})$ given Y. If we assume that the distribution of the labels Y has bounded support S, that the family $\{F_y(t)|y \in S\}$ has stable cross-tail variability, and that the shape parameters ξ_y of $F_y(t)$ change continuously, then the tail shape parameters of $W_{\boldsymbol{V}}(\boldsymbol{U})$ and $|\hat{f}_{\boldsymbol{V}}(\boldsymbol{X})|^p$ share the same sign, and are identical if either of them is positive.*

There exists a strong connection between the Maximum Domain of Attraction of a distribution, and the existence of its moments (see [Embrechts, 2013]):

Proposition 4.4.15. *If $F_{|X|}$ is the distribution function of a random variable $|X|$, and $F_{|X|} \in MDA(\xi)$ then:*

$$\textit{i) if } \xi > 0, \textit{ then } \mathbb{E}[|X|^r] = \infty, \forall r \in (\frac{1}{\xi}, \infty), \tag{4.28}$$

$$\textit{ii) if } \xi \leq 0, \textit{ then } \mathbb{E}[|X|^r] < \infty, \forall r \in (0, \infty). \tag{4.29}$$

This means that, for a model with a positive loss function whose distribution has a shape parameter that is bigger than one, even the first moment of that loss function distribution does not exist. Hence, we would expect that our model has an infinite mean, which would suggest that this model should be eliminated during model ranking. However, if every model has an infinite expected loss, it's not advisable to eliminate them all. An alternative approach could be to utilize the tail thickness and medians of the loss function distributions to guide decision-making about which models to keep.

In Proposition 4.4.14, we showed that if we condition on the testing set, under some assumptions, we can estimate the shape of the total loss distribution, that is the distribution of $W_{\boldsymbol{V}}(\boldsymbol{U})$, by simply investigating the models prediction, without the need for target data. This can also be motivated from the moments of $W_{\boldsymbol{V}}(\boldsymbol{U})$ as shown in Appendix C.4.

4.5 Experiments

In this section, we demonstrate the significance of Theorem 4.4.11. In the first subsection, we show experimental evidence that the estimated shape parameter of the marginal distribution, coincides with the maximal shape parameter of individual conditional distributions. In the second subsection, we show that when the sample size is finite, as it is the case in practice, the method proposed by Theorem 4.4.11 (cross tail estimation) can be necessary to reduce the required sample size for proper tail shape parameter estimation of marginal distributions. Furthermore, in the third subsection, we compare the standard POT and cross tail estimation on real data. For the considered regression scenarios, we notice that when these shape parameters are calculated by cross tail estimation, the magnitude of shape parameters of the distribution of model predictions increases significantly when the model overfits. We also notice that such a relationship does not hold in the case that we use directly the POT method to estimate the aforementioned shape parameters. Finally, in the fourth subsection, we discuss the computational advantages of using cross tail estimation.

4.5.1 Validity of Cross Tail Estimation in Practice

The main problem that we tried to tackle in the previous section was estimating the shape parameters of the tail of distribution $F(x)$:

$$F(x) = \int f(\boldsymbol{z}) F_{\boldsymbol{z}}(x) d\boldsymbol{z}, \tag{4.30}$$

via tail shape estimation of the conditional distributions $F_{\boldsymbol{z}}(x)$. In what follows, we provide experiments showing that this is feasible in practice. **Experimental Setting**

For simplicity, we set $\boldsymbol{z}$ to be one dimensional, and thus denote the conditional distributions $F_{\boldsymbol{z}}$ as F_z, where $z \in \mathbb{R}$. In this case Equation (4.30) becomes

$$F(x) = \int f(z) F_z(x) dz. \tag{4.31}$$

First, we define $f(z)$ as a mixture of Gaussian distributions. To do so we choose a mean μ_i from a uniform distribution in $[-5, 5]$ and then a standard deviation σ_i from a uniform distribution between $[0, 4]$, together defining a Gaussian distribution $g_i(z)$. We repeat this process for 30 Gaussian distributions and define $f(z) = \sum_{i=1}^{30} \frac{g_i(z)}{30}$.

Second, we define the function ξ_z as

$$\xi_z = \frac{\frac{(nz+2m^2+kz^3)e^{-|z|}+a}{b} + c}{d}, \tag{4.32}$$

where $n = 1$, $m = 2$, $k = 2$, $b = 5.76$, $a = -3b - 3.80$, $d = (\frac{7}{8}\xi_{\max} + \frac{29}{8})^{-1}$ and $c = d\xi_{\max} + 3$. The $\xi_{\max}$ in the variables c, d determines the maximum value that the function ξ_z takes as long as $\xi_{\max} \in [-4, 5]$. More details about the function ξ_z are provided in Appendix C.7.

Third, we define $F_z(x)$. If $\xi_z \leq 0$, then we define $F_z(x)$ as a Generalized Pareto distribution (GPD) where the scale parameter is set to 1 and the tail shape parameter is ξ_z. Otherwise we define $F_z(x)$ as $1 - x^{-\frac{1}{\xi_z}}$. The choice of $\xi_{\max}$ completely determines each ξ_z and hence each $F_z(x)$, thus it fully defines $F(x)$ in Equation (4.31).

We run the experiments for different values of the parameter $\xi_{\max}$, that is, $\xi_{\max}$ takes the following 45 values $\{-4, -4 + 0.2, -4 + 0.4, ..., 5\}$. We denote these $\xi_{\max}$ values as $\xi_j = -4 + \frac{2j}{10}$, where $j \in \{0, ..., 45\}$. Each choice of j defines a particular maximal value $\xi_{\max} = \xi_j$ and thus a marginal distribution $F_j(x)$ as on the left side of Equation (4.31). Also since the particular choice j of the maximum $\xi_{\max}$ determines all corresponding ξ_z in Equation (4.32) then we denote ξ_z as $\xi_{z,j}$.

For each j we repeat p times Algorithm 6. On repetition k, the algorithm returns $\hat{\xi}_j^k$ which is an estimation of ξ_j. As guided by the ideas laid in Appendix C.5, our final estimation of ξ_j after p repetitions of Algorithm 6 above is $\hat{\xi}_j = \frac{1}{p}\sum_{k=1}^{p}\hat{\xi}_j^k$.

Algorithm 6 Construction of a Continuous Mixture Distribution and Direct POT Usage

define: $J = \{\}$
fix the number of iterations: $M \in \mathbb{N}$
repeat
  a) Sample a z from the distribution $f(z)$
  b) For that z, calculate $\xi_{z,j}$ (given that $\xi_{\max} = \xi_j$)
  if $\xi_{z,j} \leq 0$ **then**
    c) Sample a point x from a GPD with location zero, $\sigma = 1$, shape parameter $\xi_{z,j}$.
    d) $J = J \cup \{x\}$
  else
    c) Sample x from $F_z(x) = 1 - x^{-\frac{1}{\xi_{z,j}}}$
    d) $J = J \cup \{x\}$
  end if
until $|J| = M$
apply the Pickands or DEdH estimator on J to estimate ξ_j, the shape parameter of $F_j(x)$.
return the estimated value of ξ_j

Experimental validation of CTE using the Pickands estimator

We show the results of the experiment described above, when the Pickands Estimator is applied. In this study, a comprehensive set of experimental outcomes has been illustrated in Figure 4.1. Here, the parameter M, delineated in the preceding subsection, is assigned values from the set $\{10^5, 10^6, 10^7, 10^8\}$. In the context of these experiments, p was set to 10 as a constant across all trials. The experiments were performed encompassing a total of 10 runs to capture potential variability and better reflect the stochastic nature of the process.

In order to acquire a more robust and representative understanding of the results, given the inherent variability of the experimental setup, statistical metrics including the mean and standard deviation were computed across these multiple experimental runs.

Upon examining the obtained results, they seem to align with our initial theoretical expectations. Specifically, when the maximum tail shape parameter within the mixture of conditional distributions is of positive value, the estimated shape parameter of the marginal distribution is that identical positive value. On the other hand, if the maximum tail shape parameter within the conditional distributions is negative, the estimated shape parameter of the marginal distribution duly returns a negative value. This symmetry in the estimations provides a degree of confidence in the validity of the conducted experiments and the consistency of the underlying theoretical framework.

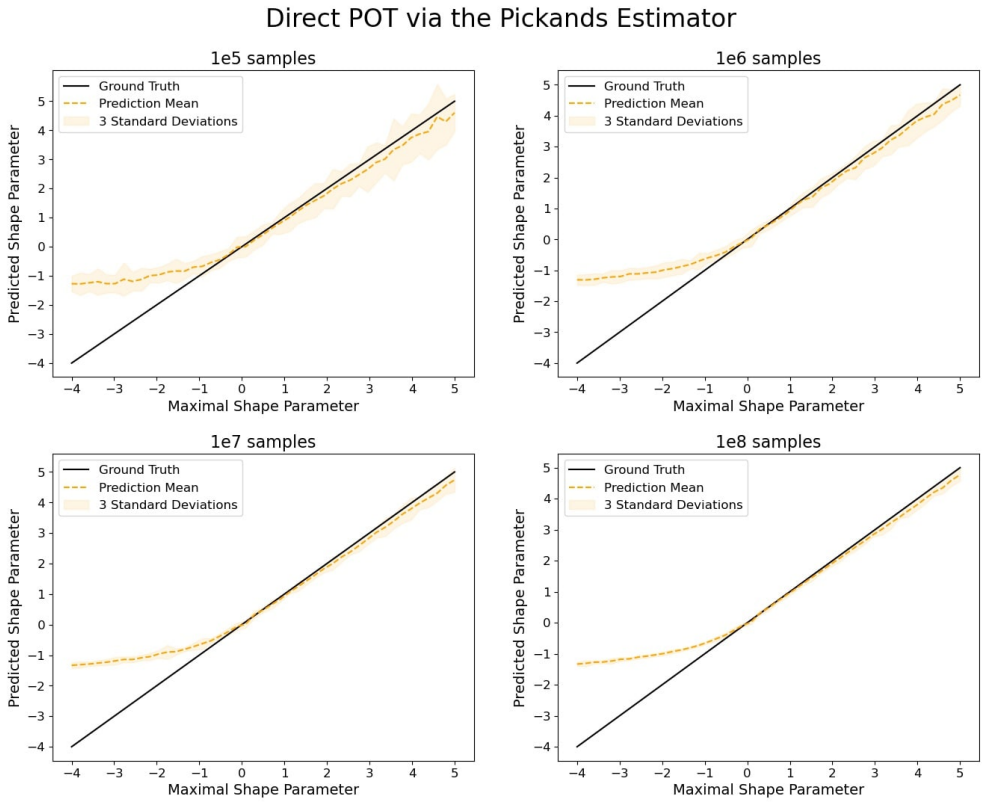

Fig. 4.1.: In cases where the maximum tail shape parameter in the mixture of conditional distributions is positive, the estimated shape parameter of the marginal is equal to this maximal value. If this maximum value is negative, the estimated shape parameter is negative. These results were obtained using the Pickands estimator.

Complementary results, pertaining to a replica of the above-described experiment, wherein the DEdH Estimator is utilized, can be found in Appendix C.9.

4.5.2 Robustness to Variance in the Location of Conditional Distributions

In subsection 4.5.1, we presented empirical evidence to substantiate Theorem 4.4.11. Notably, for computational expediency, we elected to set all conditional distributions with a location parameter of zero. This decision was motivated by the fact that, if location parameters were permitted to exhibit significant variability, the direct Peaks Over Threshold (POT) approach would necessitate an unfeasibly large sample size to verify our claims. This issue is addressed in the current subsection, wherein we illustrate that the CTE approach provides a suitable remedy. Specifically, in subsection 4.5.2, we outline modifications to the experimental setup from subsection 4.5.1 that allow for variation in the location parameter, and present the experimental results accordingly. In subsection 4.5.2, we apply the CTE approach to the same marginal distributions as in subsection 4.5.2, and demonstrate that it allows for correct estimation

of shape parameters. Additional experiments, in more simplified settings, highlighting the necessity of CTE are provided in Appendix C.6.

Applying POT directly when the location of conditional distributions exhibits substantial variability

In order to ensure high variability of the location of conditional distributions $F_z(x)$, we modify step (c) in the *if* statement of Algorithm 6 into (c*) as delineated in Algorithm 7.

Algorithm 7 Modification of Algorithm 3 to Ensure High Location Variability

if $\xi_{z,j} \leq 0$ **then**
  c) Sample a point x from a GPD with location zero, $\sigma = 1$, shape parameter $\xi_{z,j}$.
  d) $J = J \cup \{x\}$
else
  c*) Sample x from $F_z(x) = 1 - x^{-\frac{1}{\xi_{z,j}}}$. Translate x by adding $\frac{1}{\xi_{z,j}{}^4}$, i.e., $x = x + \frac{1}{\xi_{z,j}{}^4}$.
  d) $J = J \cup \{x\}$
end if

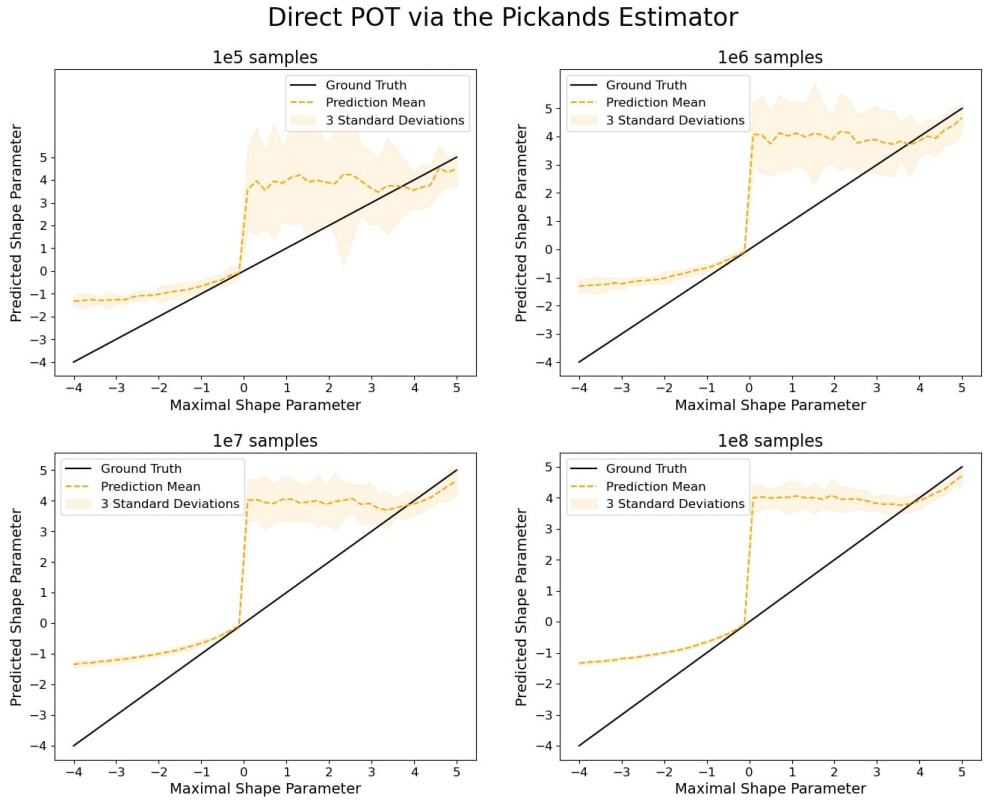

Fig. 4.2.: Estimation of the shape parameter of the marginal by direct application of POT. We utilize the Pickands estimator as our estimator of choice.

This adaptation ensures that conditional distributions with lower positive shape parameters are situated at greater distances from the origin, thereby augmenting the probability that their tails will dominate over those that exhibit heavier tails.

The results in Figure 4.2 show that the estimators predict that the shape parameter of the tail is constantly 4 as the tail of the marginal is determined by ${\xi_{z,j}}^{-4}$ instead of $1 - x^{-\frac{1}{\xi_{z,j}}}$ which merely becomes noise around $\xi_{z,j}^{-4}$. This changes once $\xi_{\max} = \xi_j$ becomes larger than 4, in which case the tails of the conditional distribution are once again determined by $1 - x^{-\frac{1}{\xi_{z,j}}}$.

Complementary results, pertaining to a replica of the above-described experiment, wherein the DEdH Estimator is utilized, can be found in Appendix C.9, Figure C.10.

Enhancing parameter estimation accuracy through the CTE approach

We demonstrate that the CTE method can effectively recover the true shape of the tail of the marginal, even in cases where the conditional distributions exhibit highly varying locations, as was observed in subsection 4.5.2. To ensure objectivity, we define the functions $f(z)$, $\xi_{z,j}$, and $F_{z,j}(x)$ in a consistent manner as before, thereby ensuring that all marginal distributions under consideration are equivalent to those studied in previous cases. As per the definition of the CTE, the sampling and estimation procedure is described in detail in Algorithm 8.

Algorithm 8 Application of CTE on the Mixture Distribution Defined in Algorithm 4

sample K values z from $f(z)$
for each z **do**
  Calculate $\xi_{z,j}$ (given that $\xi_{\max} = \xi_j$)
  for $l = 1$ to p **do**
    if $\xi_{z,j} \leq 0$ **then**
      Sample a set S of N samples from a GPD with shape parameter $\xi_{z,j}$, scale $\sigma = 1$.
    else
      Sample a set S of N samples from $F_z(x) = 1 - x^{-\frac{1}{\xi_{z,j}}}$.
      Translate each sample x in S as follows: $x = x + \frac{1}{{\xi_{z,j}}^4}$.
    end if
    Apply the Pickands or DEdH estimator on S to get an estimate $\hat{\xi}_{z,j}^{l}$ of the shape parameter $\xi_{z,j}$ of $F_{z,j}(x)$
  end for
  As guided by the ideas laid in Appendix C.5, our final estimation of $\xi_{z,j}$ after p repetitions of the process above is $\hat{\xi}_{z,j} = \frac{1}{p}\sum_{l=1}^{p} \hat{\xi}_{z,j}^{l}$.
end for
We select the maximal $\hat{\xi}_{z,j}$ from the K predicted values (corresponding to the K sampled z). According to Theorem 4.4.11 the estimated $\hat{\xi}_j$ should be close to ξ_j.
return $\hat{\xi}_j$, the estimated value of ξ_j

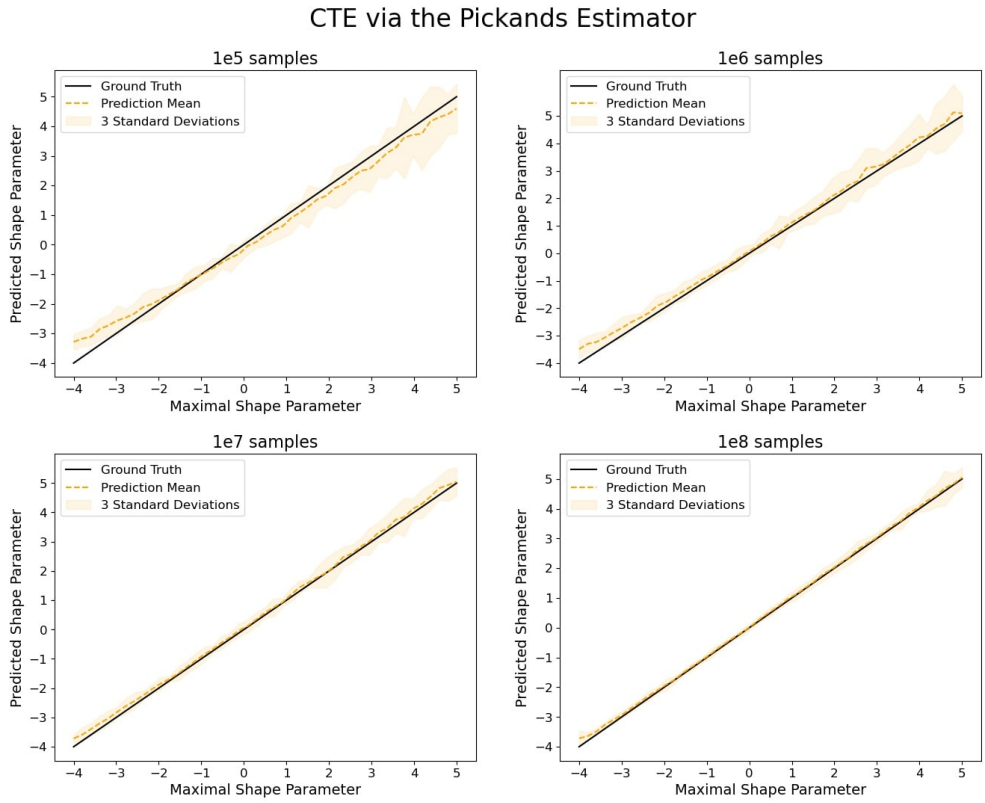

Fig. 4.3.: Estimation the shape parameter of the marginal using CTE. We utilize the Pickands estimator as our estimator of choice.

We set $p = 10$ at all times. Furthermore, for the sake of fairness, we sample the same number of points from each marginal distribution as in the previous subsection, that is, we set $KN = M$. Since we set $K = 50$, in order for M to take values in $\{1e5, 1e6, 1e7, 1e8\}$, N needs to take values in $\{2e3, 2e4, 2e5, 2e6\}$. We execute the experiment 10 times, and to account for variability across the different runs, we compute the mean and standard deviation of the results. They are shown in Figure 4.3. Naturally, the more K is increased the more likely we are to sample the z corresponding to the conditional distribution with the maximal shape parameter. Theorem 4.4.11 provides assurance that as the value of K increases, our estimation progressively converges to the true shape parameter of the marginal distribution. The conducted experiments indicate that merging samples from various conditional distributions, which form the marginal, may potentially be detrimental when estimating the tail shape of the marginal.

Complementary results, pertaining to a replica of the above-described experiment, wherein the DEdH Estimator is utilized, can be found in Appendix C.9, Figure C.11.

4.5.3 Model performance inference improvements via cross tail estimation, relative to POT

In what follows, we show that cross tail estimation can improve the estimation of the shape of the tail in realistic settings. Furthermore, we observe that in these cases, the thickness of the tail is positively correlated with over-fitting, therefore inference regarding the performance of the model is improved when using CTE instead of POT. **Gaussian Processes** In this experiment, our data is composed of a one-dimensional time series taken from the UCR Time Series Anomaly Archive [2] [Wu, 2020], which we reorganize in windows of size 2, and use each window to fit a Gaussian process (GP) model in order to predict the next value in the series. Our complete dataset D is composed of $n = 1e4$ windows. On each run we randomly select 340 points of D for training (denote D_i), and

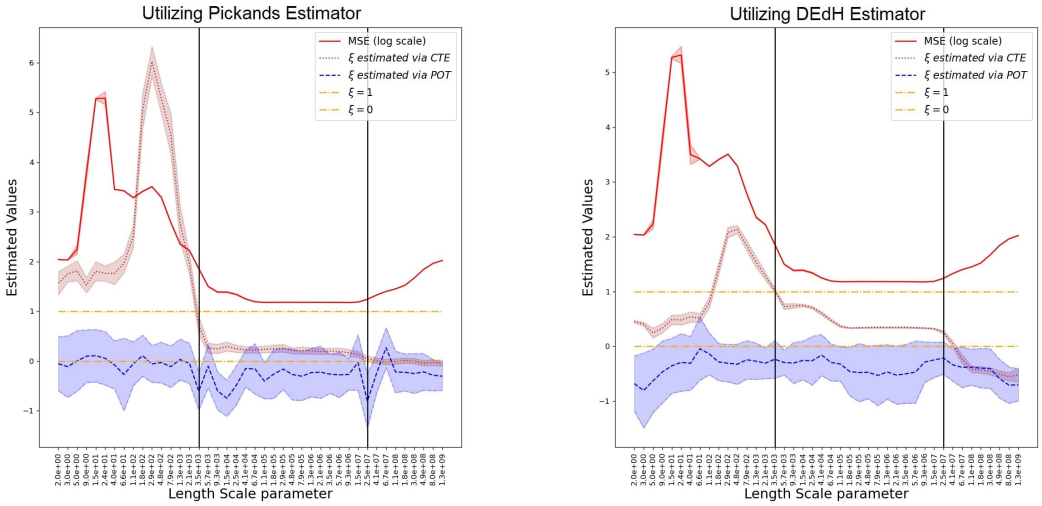

Fig. 4.4.: Experimental results in the case of testing Gaussian processes. Left: The Pickands estimator is used. Right: The DEdH estimator is used. In both cases we notice that CTE estimates larger shape parameters of the loss function distributions for models which overfit. This is not the case when POT is applied directly. The first black vertical line marks the first model with lower MSE than the model with the smallest length scale parameter (the point where the models stop overfitting). The second black vertical line marks the model in from which MSE starts growing again (the point when models begin underfitting). The MSE is presented in log scale and has been further linearly scaled to fit the plot. Shaded areas denote the standard deviation of the measurements across 200 independent runs.

then group the predictions of the model on the $1e4$ points of D into an array which we denote by $\hat{Y}_i$. Then we split $\hat{Y}_i$ into five equally sized subsets $\hat{Y}_{i,j}$. We proceed to estimate the shape parameter of the tails of the prediction of the model, for given training set D_i. This is done by applying the Pickands/DEdH estimator to $\hat{Y}_{i,j}$, receiving $\hat{\xi}_{i,j}$ and then as per Appendix C.5, we get the estimate $\hat{\xi}_i = \frac{1}{5}\sum_{j=1}^{5}\hat{\xi}_{i,j}$ which corresponds to $\hat{Y}_i$. We

[2] `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/UCR_TimeSeriesAnomalyDatasets2021.zip`

repeat this process 1000 times (for 1000 choices of the training set D_i), and select as our estimation of the shape parameter of the tail of the distribution of our loss function, the maximum individual estimated parameter: $\hat{\xi}_i = \max\{\hat{\xi}_i | i \in [1000]\}$. On the other hand, we also calculate the MSE on the testing set $D \setminus D_i$ after the model has been trained on D_i.

To check the difference of performance of the direct POT of tail shape estimation and cross tail estimation, we also calculate the shape parameter of the overall distribution of prediction models, through the standard method, by applying Pickands/DEdH estimator on $Y = \bigcup_{i=1}^{1000} \hat{Y}_i$.

These experiments are repeated for length scale parameters given in the $x-$axis of Figure 4.4 as well as in Appendix C.8. We repeat every experiment 200 times to account for variability across different runs, we compute the mean and standard deviation of the results.

In Figure 4.4 , we notice that when the CTE approach is used, the shape parameter is significantly larger for models which overfit. In Appendix C.8, we illustrate that the MSE is large for small scale parameters due to overfitting (Figure C.4). Furthermore, the shape parameter only drops to (under) zero, when the model starts underfitting for length scale parameters bigger than $2.5e7$. In Appendix C.8 (Figure C.5), it is shown that for such large values of the length scale parameter, the predictions become roughly constant.

On the other hand, if POT is applied directly, then the estimated shape parameters are not significantly larger for models which overfit compared to those that do not. This is because conditioning on the training set, the predicted values on the test set vary significantly with regards to the their location. Hence, the tail that is estimated by the direct application of the POT approach is sometimes simply the one translated the furthest from the origin. Thus, if there is some inverse relationship between the magnitude of the location and the size of the estimated shape parameter across different conditional, then we expect POT to underestimate the true shape parameter of the marginal. This is shown in Appendix C.8, for the model with the highest estimated shape parameter (290). The variability (sorted) of the estimated shape parameters of the 1000 conditionals for each length scale parameter is given in Figure C.7 of Appendix C.8, together with the corresponding $97th$ percentile (threshold) from each corresponding conditional distribution. We notice that indeed, quite often the difference between locations is large, and that the largest threshold often corresponds to conditional distributions with small, even negative shape parameters.

The outcomes presented herein are robust with regards to the choice of applying the method explicated in Appendix C.5 in conjunction with the direct Peaks Over Threshold

(POT) approach. Furthermore, the findings presented in Figure 4.4 demonstrate near equivalence in relation to the magnitude of the selected threshold (in this study, we evaluated 99.7 and 99.997 percentiles).

Polynomial Kernels This experiment is almost identical to the previous one, with the only differences being that the models we test now are polynomial kernels, and the set of possible candidate models in this case is defined by the degree of the polynomial kernel. We test polynomial kernels of degree from 1 to 9. As before, we repeat this experiment 200 times. The results are shown in Figure 4.5.

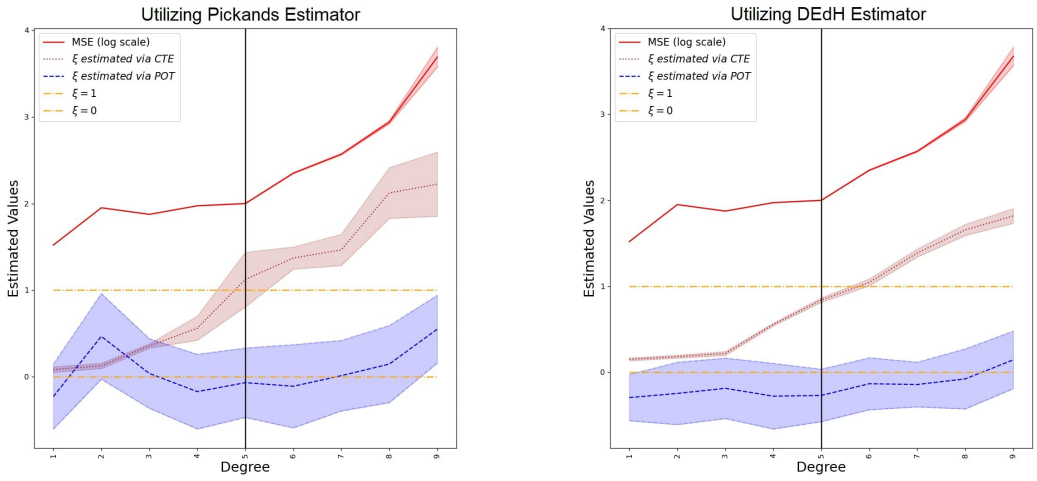

Fig. 4.5.: Experimental results in the case of testing polynomial kernels. Left: The Pickands estimator is used. Right: The DEdH estimator is used. In both cases we notice that CTE estimates larger shape parameters of the loss function distributions for models which overfit. This is not the case when POT is applied directly. The black vertical line marks the inflection point of the MSE. The MSE is presented in log scale and has been further linearly scaled to fit the plot. Shaded areas denote the standards deviation of the measurement across 200 independent runs.

4.5.4 Computational Simplifications

Another benefit to using cross tail estimation is the reduction of computational time, as for a given number m of conditional distributions, with n samples for each, instead of joining all testing samples together in an array of size $m * n$, we perform calculations in m arrays of size n in parallel. This becomes useful in practice during shape parameter estimation, as using Pickands estimators requires sorted samples, where best algorithms for sorting require $n \log(n)$ operations for a vector of size n. Hence our method which requires $n \log(n)$ operations is much faster in practice than the standard POT approach which requires $mn \log(mn)$, in a setting where m and n are of approximately of the same order.

4.6 Conclusion

We study the problem of estimating the tail shape of loss function distributions, and explain the complications that arise in performing this task. We notice that such complications arise in general during the estimation of the tail shape of marginal distributions. In order to mitigate such shortcomings, we propose a new method of estimating the shape of the right tails of marginal distributions and give theoretical guarantees that the tail of the marginal distribution coincides with the thickest tail of the set of conditional distributions composing the marginal. We give experimental evidence that our method works in practice, and is necessary in applications with small sample sizes. Using the aforementioned method, we show experimentally that the tails of distribution functions in many cases can have non-exponential decay, as well as that it is possible that not even their first moment exists. Furthermore, we discover an interesting phenomena regarding the relationship between the overfitting of a model, and the thickness of the tails of its prediction function distribution, in the experiments we conducted.

Potential additional applications of the method we develop include improving classic tail modelling, as well as the threshold selection for model comparison in anomaly detection [Su, 2019]. Furthermore, cross tail estimation could be used to estimate the existence of the moments of loss function distributions, and thus can be considered as a potential elimination criteria for models whose first moment does not exist.

5 Conclusion and Perspectives

Contents

In the first part of this thesis, we focused on modelling high-dimensional, complex data distributions. More precisely, in Chapter 2, we introduced AFFJORD, a framework that guarantees accurate representation of probability density functions and enhances the performance of ODE based Normalizing Flows. In Chapter 3, we put forth a novel approach named PSM, that not only improves density estimation when compared to the standard approach of training diffusion models, but also expedites the training process, all while maintaining roughly identical inference time and memory consumption. In the second part we introduced 'cross-tail estimation,' a new method to facilitate the estimation of tail thickness in marginal distributions, exhibiting superior performance when contrasted with the direct application of Points-Over-Threshold approach.

5.1 Summary of the Main Contributions

In this section, we present a synopsis of the principal contributions.

5.1.1 Enhanced Distribution Modelling via Augmented Architectures For Neural ODE Flows

In the second chapter of this thesis, we tackled the issue of insufficient flexibility observed in the evolution of the vector field within the framework of Continuous Normalizing Flows. In response to this issue, we endeavored to develop an effective solution, resulting in the contributions listed below.

Main Contributions

- We propose an intuitive expansion of the total derivative formula in the continuous sense, demonstrating its equivalence with continuous backpropagation.

- We introduce the continuous version of the chain rule, which we refer to as the cable rule. The cable rule bears resemblance to the forward sensitivity of ordinary differential equations (ODEs) as it details the evolution of the state's Jacobian based on initial conditions, akin to how forward sensitivity illustrates the Jacobian's evolution in relation to flow parameters.

- Drawing inspiration from our results, we put forth a novel type of continuous normalizing flow known as Augmented FFJORD (AFFJORD). This method surpasses the currently leading continuous normalizing flow method, FFJORD, in our experimental findings.

5.1.2 Faster Training of Diffusion Models and Improved Density Estimation via Parallel Score Matching

In the third chapter, we focused on diffusion probabilistic models. We directed our efforts toward refining training methodologies, with a specific focus on Parallel Score Matching strategies. The motivation behind this exploration was to harness the inherent property of diffusion models, which enable the individualized modeling of distinct scores. We list our main contributions below.

Main Contributions

- Capitalizing on the inherent attribute of diffusion models, which allows for the individual modeling of each score, we present evidence supporting the feasibility of this method for density estimation.

- Our proposed method not only enhances the outcomes in density estimation but also reduces training duration through the process of parallelization.

- We provide empirical evidence to assert that our approach does not incur penalties in terms of inference time and memory expenses.

- We further demonstrate that in specific instances, our approach improves the generative process.

5.1.3 On Tail Decay Rate Estimation of Loss Function Distributions

In the fourth chapter of the thesis, drawing inspiration from cross-validation and the process of marginalizing the loss function across various training sets in machine learning models, we directed our focus towards addressing the challenge of enhancing the estimation of the tail shape parameter in the case of general marginal distributions. In the pursuit of this objective, we made several contributions which are provided below.

Main Contributions

- We tackle the task of measuring the tail shape of loss function distributions and introduce a new method to estimate the shape of the tails of general marginal distributions. This approach offers theoretical guarantees that the tail of the marginal distribution aligns with the thickest tail among the underlying conditional distributions.

- We provide empirical evidence to affirm the efficacy of our method in real-world applications, especially in scenarios dealing with limited sample sizes, and show that tails of loss-function distributions often show non-exponential decay.

- We expose an interesting link between the overfitting of a model and the thickness of the tail of its corresponding prediction-function distribution.

- Our developed method has potential implications for improving classic tail modeling, aiding in threshold selection for model comparison in anomaly detection, and may serve as a criterion to eliminate models whose first moment of its loss function distribution does not exist.

- In the task of tail shape estimation of marginal distributions, the method we developed provides computational advantages when the number of conditional distributions is of the same order as the number of samples from each conditional.

5.2 Perspectives and Future Applications

In the ensuing discussion, we offer a detailed perspective along with potential pathways for future applications.

5.2.1 Fine-tuning of Diffusion Models for Density Estimation

Time series data are often non-stationary, altering their distribution over different periods. Consequently, learning the distribution of such time series, particularly for anomaly detection using a singular diffusion model, can pose significant challenges. Recently, numerous architectural alterations and augmentations, such as LoRA and ControlNet [Hu, 2021; Zhang, 2023], have been proposed, enabling the fine-tuning of diffusion models to cater to specific distributions.

Future research could be directed towards designing models capable of discerning the current state of the time series and dynamically adjusting the model via control strategies as mentioned above. The implementation of such a model would involve the continuous detection and adaptation to the evolving states of the time series data, providing a real-time response to changing conditions. This proposed framework bears resemblance to high-capability, multi-regime Markov Switching Models [Hamilton, 2010; Guidolin, 2007]. Such a model would operate by understanding the current state of the time series and adjusting its peformance accordingly.

5.2.2 Faster Density Estimation

A key limitation of diffusion models is their requirement for multiple steps to generate a sample, subsequently necessitating a substantial number of steps to execute precise density estimation. Recent studies have sought to reduce the number of steps required for high-quality generation [Lu, 2022; Karras, 2022]. However, many of these investigations primarily concentrate on the generative task, neglecting to provide results for density estimation. Exploring the applicability of such methods to density estimation would be an intriguing prospect. Among the recent methodologies with promising outcomes are consistency models [Song, 2023], which enable high-quality single-step sampling. Regrettably, this generative function ceases to be a diffeomorphism. Uncovering methods

to ensure that such functions retain their diffeomorphic properties and are thus suitable for density estimation presents an exciting direction for future research.

5.2.3 Flexible Conditional Density Estimation

In some cases we would like to perform density estimation of a random variable, knowing the realized value of depended random variables. Diffusion models enable us to do so, as one can condition on the variable $\boldsymbol{z}$ in order to generate or estimate the likelihood of $\boldsymbol{y}$. This can be done through various ways [Dhariwal, 2021; Ho, 2022; Kim, 2022; Nichol, 2021a], most notably as in [Saharia, 2022a; Saharia, 2022b]. In this case, one trains a diffusion model on the noisy versions $\boldsymbol{y}_t$ of the original input $\boldsymbol{y}_0$, while constantly keeping as input in the network the realized value of the variable on which we condition, that is, $\boldsymbol{z}$. While this works in practice, we would like to be able to train a diffusion model on $\boldsymbol{x}$, and be able to perform conditional density estimation for any splitting $\boldsymbol{y}, \boldsymbol{z}$ of $\boldsymbol{x}$, that is $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{z})$. Naturally since the diffusion model has modelled the joint distribution $p(\boldsymbol{x})$, we know that it contains the information $p(\boldsymbol{z}) = \int p(\boldsymbol{y}, \boldsymbol{z}) d\boldsymbol{y}$, and thus $p(\boldsymbol{y}|\boldsymbol{z}) = \frac{p(\boldsymbol{x})}{p(\boldsymbol{z})}$. Strategies for generating $\boldsymbol{y}$ given $\boldsymbol{z}$ that work well in practice have been developed and applied to inpainting, colorization, class generation to name a few [Song, 2021; Lugmayr, 2022]. However, there is no guarantee that the generated samples are actually samples from $p(\boldsymbol{y}|\boldsymbol{z}) = \frac{p(\boldsymbol{x})}{p(\boldsymbol{z})}$. More importantly, in the context of this thesis, no methods exist to perform density estimation of $p(\boldsymbol{y}|\boldsymbol{z})$, if one has a diffusion model trained on $p(\boldsymbol{x}) = p(\boldsymbol{y}, \boldsymbol{z})$.

5.3 Final Remarks

In this thesis, we have delved into the significance of density estimation and explored a variety of both classical and contemporary methods dedicated to this area. Additionally, we have examined the tails of distributions and looked at a collection of concepts and techniques used in Extreme Value Theory (EVT), a discipline that focuses on the distribution of extreme values.

Throughout our discussion, we highlighted numerous challenges that emerge within these topics, and presented the efforts we've made to alleviate these issues. Our contributions enabled us to enhance the performance of contemporary methods for density estimation, and improved the estimation of the shape of tails marginal distributions.

Towards the conclusion, we offered several perspectives for future research. This includes ways for accelerating diffusion models, strategies for efficient fine-tuning of such methods, and potential for transforming models trained to estimate joint density functions into ones capable of estimating conditional densities derived from the joint one.

These perspectives present intriguing possibilities, particularly for anomaly detection in non-stationary time series and in the enhancement of regression procedures. Such advancements could, in turn, find practical applications in a wide array of real-world tasks. As we close this thesis, we look forward to seeing the progress and innovations that will undoubtedly continue to emerge in this exciting and ever-evolving field.

Bibliography

[Adams, 2018] Ryan P. Adams, Jeffrey Pennington, Matthew J. Johnson, et al. “Estimating the Spectral Density of Large Implicit Matrices”. In: *ArXiv Preprint* (2018) (cit. on pp. 7, 22).

[Akaike, 1973] Hirotogu Akaike. “Information Theory and an Extension of the Maximum Likelihood Principle”. In: *Selected Papers of Hirotugu Akaike*. New York, NY: Springer New York, 1973, pp. 199–213 (cit. on p. 54).

[Akaike, 1974] H. Akaike. “A new look at the statistical model identification”. In: *IEEE Transactions on Automatic Control* 19.6 (1974), pp. 716–723 (cit. on p. 54).

[Anderson, 1980] G. Leigh Anderson and Rui J. P. de Figueiredo. “An Adaptive Orthogonal-Series Estimator for Probability Density Functions”. In: *The Annals of Statistics* (1980) (cit. on p. 4).

[Arlot, 2009] Sylvain Arlot and Pascal Massart. “Data-driven Calibration of Penalties for Least-Squares Regression”. In: *Journal of Machine Learning Research* 10.10 (2009), pp. 245–279 (cit. on p. 54).

[Balkema, 1974] A. A. Balkema and L. de Haan. “Residual Life Time at Great Age”. In: *The Annals of Probability* 2.5 (1974), pp. 792–804 (cit. on pp. 11, 58).

[Behrmann, 2019] Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Joern-Henrik Jacobsen. “Invertible Residual Networks”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 573–582 (cit. on p. 39).

[Bengio, 1999] Yoshua Bengio and Samy Bengio. “Modeling High-Dimensional Discrete Data with Multi-Layer Neural Networks”. In: *Advances in Neural Information Processing Systems*. MIT Press, 1999 (cit. on p. 3).

[Biernacki, 2003] Christophe Biernacki, Gilles Celeux, and Gérard Govaert. “Choosing Starting Values for the EM Algorithm for Getting the Highest Likelihood in Multivariate Gaussian Mixture Models”. In: *Comput. Stat. Data Anal.* (2003) (cit. on p. 2).

[Birge, 1995] Lucien Birge and Pascal Massart. “Estimation of Integral Functionals of a Density”. In: *The Annals of Statistics* 23.1 (1995), pp. 11–29 (cit. on p. 54).

[Bishop, 2007] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007 (cit. on p. 2).

[Blanes, 2009] S. Blanes, F. Casas, J.A. Oteo, and J. Ros. "The Magnus Expansion and Some of its Applications". In: *Physics Reports* (2009) (cit. on pp. 25, 101).

[Bond-Taylor, 2021] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G Willcocks. "Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models". In: *IEEE transactions on pattern analysis and machine intelligence* (2021) (cit. on p. 5).

[Box, 1994] George E. P. Box, Gregory C. Reinsel, and Gwilym M. Jenkins. *Time series analysis : forecasting and control*. Prentice-Hall, 1994 (cit. on p. 3).

[Burnham, 2007] K.P. Burnham and D.R. Anderson. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer New York, 2007 (cit. on p. 54).

[Casella, 2001] George Casella and Roger Berger. *Statistical Inference*. Duxbury Resource Center, 2001 (cit. on p. 2).

[Chen, 2018] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. "Neural Ordinary Differential Equations". In: *Advances in Neural Information Processing Systems*. 2018 (cit. on pp. 7, 8, 20, 22, 26, 27, 38, 105).

[Cochran, 2007] W.G. Cochran. *Sampling Techniques, 3Rd Edition*. A Wiley publication in applied statistics. Wiley India Pvt. Limited, 2007 (cit. on p. 55).

[Davis, 1984] Richard Davis and Sidney Resnick. "Tail Estimates Motivated by Extreme Value Theory". In: *The Annals of Statistics* 12.4 (1984), pp. 1467–1487 (cit. on p. 140).

[Dekkers, 1989a] A. L. M. Dekkers, J. H. J. Einmahl, and L. De Haan. "A Moment Estimator for the Index of an Extreme-Value Distribution". In: *The Annals of Statistics* 17.4 (1989), pp. 1833–1855 (cit. on pp. 11, 59, 140).

[Dekkers, 1989b] Arnold L. M. Dekkers and Laurens De Haan. "On the Estimation of the Extreme-Value Index and Large Quantile Estimation". In: *The Annals of Statistics* 17.4 (1989), pp. 1795–1832 (cit. on p. 140).

[Deng, 2009] Jia Deng, Wei Dong, Richard Socher, et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255 (cit. on p. 37).

[Dhariwal, 2021] Prafulla Dhariwal and Alexander Nichol. "Diffusion models beat gans on image synthesis". In: *Advances in neural information processing systems* (2021) (cit. on p. 85).

[Dinh, 2015] Laurent Dinh, David Krueger, and Yoshua Bengio. "NICE: Non-linear Independent Components Estimation". In: *Workshop paper, International Conference on Learning Representations*. 2015 (cit. on pp. 6, 21, 38).

[Dinh, 2017] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density Estimation using Real NVP". In: *International Conference on Learning Representations*. 2017 (cit. on pp. 7, 20, 22, 31).

[Dupont, 2019] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. “Augmented Neural ODEs”. In: *Advances in Neural Information Processing Systems*. 2019 (cit. on pp. 20, 23, 24, 38).

[Embrechts, 2013] P. Embrechts, C. Klüppelberg, and T. Mikosch. *Modelling Extremal Events: for Insurance and Finance*. Stochastic Modelling and Applied Probability. Springer Berlin Heidelberg, 2013 (cit. on pp. 59, 69).

[Everitt, 1981] Brian Everitt and D. J Hand. *Finite mixture distributions*. Chapman and Hall, 1981 (cit. on p. 2).

[Ferreira, 2015] Ana Ferreira and Laurens de Haan. “On the block maxima method in extreme value theory: PWM estimators”. In: *The Annals of Statistics* (2015) (cit. on p. 11).

[Finlay, 2020] Chris Finlay, Joern-Henrik Jacobsen, Levon Nurbekyan, and Adam Oberman. “How to Train Your Neural ODE”. In: *International Conference on Machine Learning*. 2020 (cit. on p. 33).

[Fisher, 1928] R. A. Fisher and L. H. C. Tippett. “Limiting forms of the frequency distribution of the largest or smallest member of a sample”. In: *Proceedings of the Cambridge Philosophical Society* 24.2 (Jan. 1928), p. 180 (cit. on pp. 10, 57).

[Galambos, 1973] J. Galambos and E. Seneta. “Regularly Varying Sequences”. In: *Proceedings of the American Mathematical Society* 41.1 (1973), pp. 110–116 (cit. on p. 59).

[Gnedenko, 1943] B. Gnedenko. “Sur La Distribution Limite Du Terme Maximum D’Une Serie Aleatoire”. In: *Annals of Mathematics* 44.3 (1943), pp. 423–453 (cit. on pp. 10, 57).

[Goh, 2016] Jonathan Goh, Sridhar Adepu, Khurum Junejo, and Aditya Mathur. “A Dataset to Support Research in the Design of Secure Water Treatment Systems”. In: Oct. 2016 (cit. on p. 125).

[Goodfellow, 2016] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016 (cit. on pp. 2, 5).

[Grathwohl, 2019] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. “FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models”. In: *International Conference on Learning Representations*. 2019 (cit. on pp. 7, 20, 22, 28, 31, 32, 47).

[Guidolin, 2007] Massimo Guidolin and Allan Timmermann. “Asset allocation under multivariate regime switching”. In: *Journal of Economic Dynamics and Control* (2007) (cit. on p. 84).

[Gumbel, 1958] E. J. Gumbel. *Statistics of Extremes*. Columbia University Press, 1958 (cit. on p. 11).

[Haan, 2007] L. de Haan and A. Ferreira. *Extreme Value Theory: An Introduction*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2007 (cit. on pp. 55, 59).

[Hamilton, 2010] James D. Hamilton. “Regime switching models”. In: *Macroeconometrics and Time Series Analysis*. Palgrave Macmillan UK, 2010 (cit. on p. 84).

[Ho, 2019] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. "Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design". In: *International Conference on Learning Representations*. 2019 (cit. on p. 32).

[Ho, 2020] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851 (cit. on pp. 9, 36, 40, 41, 45, 47).

[Ho, 2022] Jonathan Ho and Tim Salimans. "Classifier-free diffusion guidance". In: *arXiv preprint arXiv:2207.12598* (2022) (cit. on p. 85).

[Hosking, 1985] J. R. M. Hosking, J. R. Wallis, and E. F. Wood. "Estimation of the Generalized Extreme-Value Distribution by the Method of Probability-Weighted Moments". In: *Technometrics* (1985) (cit. on p. 11).

[Hu, 2021] Edward J Hu, Yelong Shen, Phillip Wallis, et al. "Lora: Low-rank adaptation of large language models". In: *arXiv preprint arXiv:2106.09685* (2021) (cit. on p. 84).

[Hurvich, 1989] Clifford M. Hurvich and Chih-Ling Tsai. "Regression and time series model selection in small samples". In: *Biometrika* 76.2 (June 1989), pp. 297–307. eprint: `https://academic.oup.com/biomet/article-pdf/76/2/297/737009/76-2-297.pdf` (cit. on p. 54).

[Hutchinson, 1990] M. F. Hutchinson. "A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines". In: *Communications in Statistics - Simulation and Computation* (1990) (cit. on pp. 7, 20, 22, 47).

[Hyvärinen, 2005] Aapo Hyvärinen. "Estimation of Non-Normalized Statistical Models by Score Matching". In: *Journal of Machine Learning Research* 6.24 (2005), pp. 695–709 (cit. on p. 36).

[Karras, 2022] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. "Elucidating the Design Space of Diffusion-Based Generative Models". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. 2022 (cit. on p. 84).

[Kim, 2022] Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. "Diffusionclip: Text-guided diffusion models for robust image manipulation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 2426–2435 (cit. on p. 85).

[Kingma, 2016] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, et al. "Improved Variational Inference with Inverse Autoregressive Flow". In: *Advances in Neural Information Processing Systems*. 2016 (cit. on p. 5).

[Kingma, 2018] Diederik P. Kingma and Prafulla Dhariwal. "Glow: Generative Flow with Invertible 1x1 Convolutions". In: *Advances in Neural Information Processing Systems*. 2018 (cit. on p. 31).

[Kingma, 2021] Diederik P Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. “On Density Estimation with Diffusion Models”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan. 2021 (cit. on pp. 9, 41).

[Kobyzev, 2021] Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. “Normalizing Flows: An Introduction and Review of Current Methods”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021) (cit. on pp. 7, 20, 22, 38).

[Kynkäänniemi, 2023] Tuomas Kynkäänniemi, Tero Karras, Miika Aittala, Timo Aila, and Jaakko Lehtinen. “The Role of ImageNet Classes in Fréchet Inception Distance”. In: *The Eleventh International Conference on Learning Representations*. 2023 (cit. on p. 115).

[Larochelle, 2011] Hugo Larochelle and Iain Murray. “The Neural Autoregressive Distribution Estimator”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. PMLR, 2011 (cit. on p. 3).

[LeCun, 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* (2015) (cit. on p. 5).

[Li, 2019] Yuelin Li, Elizabeth Schofield, and Mithat Gönen. “A tutorial on Dirichlet process mixture modeling”. In: *Journal of Mathematical Psychology* (2019) (cit. on p. 4).

[Lipman, 2023] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. “Flow Matching for Generative Modeling”. In: *The Eleventh International Conference on Learning Representations*. 2023 (cit. on pp. 48, 122).

[Liu, 2015] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. “Deep Learning Face Attributes in the Wild”. In: *Proceedings of International Conference on Computer Vision (ICCV)*. 2015 (cit. on p. 37).

[Lu, 2022] Cheng Lu, Yuhao Zhou, Fan Bao, et al. “Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps”. In: *Advances in Neural Information Processing Systems* (2022) (cit. on p. 84).

[Lugmayr, 2022] Andreas Lugmayr, Martin Danelljan, Andres Romero, et al. “Repaint: Inpainting using denoising diffusion probabilistic models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 11461–11471 (cit. on p. 85).

[Magnus, 1954] W. Magnus. “On the Exponential Solution of Differential Equations for a Linear Operator”. In: *Communications on Pure and Applied Mathematics* (1954) (cit. on pp. 25, 101).

[Massaroli, 2020] Stefano Massaroli, Michael Poli, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. “Dissecting Neural ODEs”. In: *Advances in Neural Information Processing Systems*. 2020 (cit. on p. 26).

[Meguelati, 2019] Khadidja Meguelati, Benedicte Fontez, Nadine Hilgert, and Florent Masseglia. “High Dimensional Data Clustering by means of Distributed Dirichlet Process Mixture Models”. In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019 (cit. on p. 4).

[Mehrish, 2023] Ambuj Mehrish, Navonil Majumder, Rishabh Bharadwaj, Rada Mihalcea, and Soujanya Poria. “A review of deep learning techniques for speech processing”. In: *Information Fusion* (2023) (cit. on p. 5).

[Mikosch, 1999] T. Mikosch, Operations Research EURANDOM European Institute for Statistics Probability, and their Applications. *Regular Variation, Subexponentiality and Their Applications in Probability Theory*. EURANDOM report. Eindhoven University of Technology, 1999 (cit. on p. 60).

[Neal, 2000] Radford M. Neal. “Markov Chain Sampling Methods for Dirichlet Process Mixture Models”. In: *Journal of Computational and Graphical Statistics* (2000) (cit. on p. 4).

[Neyman, 1934] Jerzy Neyman. “On the Two Different Aspects of the Representative Method: The Method of Stratified Sampling and the Method of Purposive Selection”. In: *Journal of the Royal Statistical Society* 97.4 (1934), pp. 558–625 (cit. on p. 55).

[Nichol, 2021a] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, et al. “Glide: Towards photorealistic image generation and editing with text-guided diffusion models”. In: *arXiv preprint arXiv:2112.10741* (2021) (cit. on p. 85).

[Nichol, 2021b] Alexander Quinn Nichol and Prafulla Dhariwal. “Improved Denoising Diffusion Probabilistic Models”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 8162–8171 (cit. on p. 48).

[Papamakarios, 2017] George Papamakarios, Theo Pavlakou, and Iain Murray. “Masked Autoregressive Flow for Density Estimation”. In: *Advances in Neural Information Processing Systems*. 2017 (cit. on pp. 5, 6).

[Papamakarios, 2021] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. “Normalizing Flows for Probabilistic Modeling and Inference”. In: *Journal of Machine Learning Research* (2021) (cit. on pp. 7, 22, 38).

[Parzen, 1962] Emanuel Parzen. “On Estimation of a Probability Density Function and Mode”. In: *The Annals of Mathematical Statistics* (1962) (cit. on p. 3).

[Pickands, 1975] James Pickands. “Statistical Inference Using Extreme Order Statistics”. In: *The Annals of Statistics* 3.1 (1975), pp. 119–131 (cit. on pp. 11, 55, 58).

[Pontrjagin, 1962] L.S. Pontrjagin, V.G. Boltyanskii, R.V. Gamkrelidze, E.F. Mishchenko, and D.E. Brown. *The Mathematical Theory of Optimal Processes*. International series of monographs in pure and applied mathematics. 1962 (cit. on pp. 22, 26).

[Prescott, 1980] P. Prescott and A. T. WALDEN. “Maximum likelihood estimation of the parameters of the generalized extreme-value distribution”. In: *Biometrika* (1980) (cit. on p. 11).

[Rasmussen, 1999] Carl Rasmussen. “The Infinite Gaussian Mixture Model”. In: *Advances in Neural Information Processing Systems*. 1999 (cit. on p. 4).

[Rezende, 2015] Danilo Jimenez Rezende and Shakir Mohamed. “Variational Inference with Normalizing Flows”. In: *International Conference on Machine Learning*. 2015 (cit. on pp. 6, 7, 20, 21, 38).

[Rombach, 2022] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. “High-Resolution Image Synthesis With Latent Diffusion Models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 10684–10695 (cit. on p. 36).

[Ronneberger, 2015] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi. Cham: Springer International Publishing, 2015, pp. 234–241 (cit. on pp. 45, 49).

[Saharia, 2022a] Chitwan Saharia, William Chan, Huiwen Chang, et al. “Palette: Image-to-image diffusion models”. In: *ACM SIGGRAPH 2022 Conference Proceedings*. 2022, pp. 1–10 (cit. on p. 85).

[Saharia, 2022b] Chitwan Saharia, Jonathan Ho, William Chan, et al. “Image super-resolution via iterative refinement”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022) (cit. on p. 85).

[Schwarz, 1978] Gideon Schwarz. “Estimating the Dimension of a Model”. In: *The Annals of Statistics* 6.2 (1978), pp. 461–464 (cit. on p. 54).

[Seidel, 2000] Wilfried Seidel, Karl Mosler, and Manfred Alker. “A Cautionary Note on Likelihood Ratio Tests in Mixture Models”. In: *Annals of the Institute of Statistical Mathematics* (2000) (cit. on p. 2).

[Seitzer, 2020] Maximilian Seitzer. *pytorch-fid: FID Score for PyTorch*. Version 0.3.0. 2020 (cit. on p. 49).

[Shireman, 2015] Emilie Shireman, Douglas Steinley, and Michael Brusco. “Examining the effect of initialization strategies on the performance of Gaussian mixture modeling”. In: *Behavior Research Methods* (2015) (cit. on p. 2).

[Silverman, 1986] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, 1986 (cit. on p. 4).

[Sohl-Dickstein, 2015] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 2256–2265 (cit. on pp. 8, 36, 39).

[Song, 2019] Yang Song and Stefano Ermon. “Generative modeling by estimating gradients of the data distribution”. In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 10).

[Song, 2020] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. “Sliced Score Matching: A Scalable Approach to Density and Score Estimation”. In: *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*. Ed. by Ryan P. Adams and Vibhav Gogate. Vol. 115. Proceedings of Machine Learning Research. PMLR, 2020, pp. 574–584 (cit. on pp. 9, 36, 41).

[Song, 2021] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, et al. “Score-Based Generative Modeling through Stochastic Differential Equations”. In: *International Conference on Learning Representations*. 2021 (cit. on pp. 9, 36, 40, 85, 114).

[Song, 2023] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. “Consistency models”. In: (2023) (cit. on p. 84).

[Su, 2019] Ya Su, Youjian Zhao, Chenhao Niu, et al. “Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’19. Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 2828–2837 (cit. on p. 79).

[Sugiura, 1978] Nariaki Sugiura. “Further analysts of the data by Akaike’ s information criterion and the finite corrections”. In: *Communications in Statistics - Theory and Methods* 7.1 (1978), pp. 13–26. eprint: `https://doi.org/10.1080/03610927808827599` (cit. on p. 54).

[Tabak, 2010] Esteban G. Tabak and Eric Vanden-Eijnden. “Density estimation by dual ascent of the log-likelihood”. In: *Communications in Mathematical Sciences* (2010) (cit. on pp. 6, 21).

[Tabak, 2013] E. G. Tabak and Cristina V. Turner. “A Family of Nonparametric Density Estimation Algorithms”. In: *Communications on Pure and Applied Mathematics* (2013) (cit. on pp. 6, 21, 38).

[Uria, 2013] Benigno Uria, Iain Murray, and Hugo Larochelle. “RNADE: The real-valued neural autoregressive density-estimator”. In: *Advances in Neural Information Processing Systems* 26 (2013) (cit. on p. 3).

[Van den Oord, 2016] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. “Conditional image generation with pixelcnn decoders”. In: *Advances in neural information processing systems* (2016) (cit. on p. 6).

[Vincent, 2011] Pascal Vincent. “A Connection Between Score Matching and Denoising Autoencoders”. In: *Neural Computation* 23.7 (2011), pp. 1661–1674 (cit. on p. 36).

[Wang, 2020] Phil Wang. *https://github.com/lucidrains/denoising-diffusion-pytorch*. 2020 (cit. on p. 45).

[Wu, 2020] Renjie Wu and Eamonn J. Keogh. “Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress”. In: *CoRR* abs/2009.13807 (2020). arXiv: `2009.13807` (cit. on p. 76).

[Zhang, 2019] Tianjun Zhang, Zhewei Yao, Amir Gholami, et al. “ANODEV2: A Coupled Neural ODE Framework”. In: *Advances in Neural Information Processing Systems*. 2019 (cit. on pp. 21, 24).

[Zhang, 2023] Lvmin Zhang and Maneesh Agrawala. “Adding conditional control to text-to-image diffusion models”. In: *arXiv preprint arXiv:2302.05543* (2023) (cit. on p. 84).

[Zhao, 2013] Haihua Zhao and Vincent A. Mousseau. “Extended Forward Sensitivity Analysis for Uncertainty Quantification”. In: *Nuclear Technology* (2013) (cit. on p. 25).

[Zhao, 2019] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. “Object detection with deep learning: A review”. In: *IEEE transactions on neural networks and learning systems* (2019) (cit. on p. 5).

Appendix of Chapter 2

A

Contents

A.1 Cable Rule, the Continuous Generalization of the Chain Rule

We will first assume that z is one dimensional. If $z_i = f_i(z_{i-1})$ for $i \in \{1, ..., n\}$ then by the chain rule we have

$$\frac{dz_n}{dz_0} = \frac{\partial z_n}{\partial z_{n-1}}\frac{dz_{n-1}}{dz_0} = \frac{\partial z_n}{\partial z_{n-1}}\frac{\partial z_{n-1}}{\partial z_{n-2}}...\frac{\partial z_1}{\partial z_0} = \frac{\partial f_n(z_{n-1})}{\partial z_{n-1}}\frac{\partial f_{n-1}(z_{n-2})}{\partial z_{n-2}}...\frac{\partial f_1(z_0)}{\partial z_0} \tag{A.1}$$

Now, if we assume that $z_i = z(t_i)$ is transformed more gradually as in $z_{i+1} = z_i + \epsilon f(z_i)$, and that $t_{i+1} = t_i + \epsilon$, we get that

$$\frac{dz_n}{dz_0} = \frac{\partial z_n}{\partial z_{n-1}}\frac{\partial z_{n-1}}{\partial z_{n-2}}...\frac{\partial z_1}{\partial z_0} = (I + \epsilon\frac{\partial f(z_{n-1})}{\partial z_{n-1}})(I + \epsilon\frac{\partial f(z_{n-2})}{\partial z_{n-2}})...(I + \epsilon\frac{\partial f(z_0)}{\partial z_0}) \tag{A.2}$$

We see that $z(t) = z(0) + \int_0^t f(z(\tau))d\tau$ is the limit of the previous iterative definition $z_{i+1} = z_i + \epsilon f(z_i, \boldsymbol{\theta}(t_i))$, when $\epsilon \to 0$. For simplicity, we have written $f(z_i) = f(z_i, \boldsymbol{\theta}, t_i)$. If we decide to expand equation A.2, we obtain

$$\frac{dz_n}{dz_0} = I + \sum_{i=0}^{n-1}\frac{\partial f(z_i)}{\partial z_i}\epsilon + \sum_{i=0}^{n-1}\sum_{j<i}\frac{\partial f(z_i)}{\partial z_i}\frac{\partial f(z_j)}{\partial z_j}\epsilon^2 + ... + \frac{\partial f(z_0)}{\partial z_0}...\frac{\partial f(z_n)}{\partial z_n}\epsilon^n \tag{A.3}$$

$$= I + S_1^n + S_2^n + ... + S_{n+1}^n, \tag{A.4}$$

where

$$S_2^n = \sum_{i=0}^{n-1}\sum_{j<i}\frac{\partial f(z_i)}{\partial z_i}\frac{\partial f(z_j)}{\partial z_j}\epsilon^2,\ S_3^n = \sum_{i=0}^{n-1}\sum_{j<i}\sum_{k<j}\frac{\partial f(z_i)}{\partial z_i}\frac{\partial f(z_j)}{\partial z_j}\frac{\partial f(z_k)}{\partial z_k}\epsilon^3, ... \tag{A.5}$$

We will focus on the sum S_2^n for a moment. Let us define

$$g_2^n(x, y) = \frac{\partial f(z_{t_i})}{\partial z_{t_i}}\frac{\partial f(z_{t_j})}{\partial z_{t_j}},\ \text{for } x \in [t_i, t_{i+1}], y \in [t_j, t_{j+1}]. \tag{A.6}$$

It is clear that g_2^n is the discretization of

$$g_2(t, u) = \frac{\partial f(z_t)}{\partial z_t}\frac{\partial f(z_u)}{\partial z_u},\ \text{for } t \in [0, T], u \in [0, T]. \tag{A.7}$$

We can notice that $S_2^n = \sum_{i=0}^{n-1}\sum_{j<i}\ g_2^n(t_i, t_j)\epsilon^2$ and that $g_2^n(t_i, t_j) = g_2^n(t_j, t_i)$ is symmetric, but in S_2^n, only takes values in the rectangles under the diagonal as illustrated in Figure A.3. If we decide to expand the sum S_2^n on the rectangles above the diagonal as in Figure A.4 and denote it as $\bar{S}_2^n = \sum_{i=0}^{n-1}\sum_{j\neq i}\ g_2^n(t_i, t_j)\epsilon^2$, then due to the symmetry of $g_2^n(t_i, t_j) = g_2^n(t_j, t_i)$, we deduce that $S_2^n = \frac{1}{2}\bar{S}_2^n$. The only rectangles missing in $\bar{S}_2^n$, regarding the discretization of $[0, T] \times [0, T]$, are the ones corresponding to the cases when $i = j$, which are the rectangles in the diagonal. It is important to emphasize

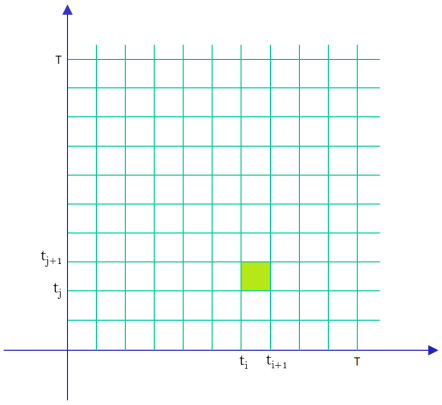

Fig. A.1.: Rectangle (i, j) in discretized $[0, T] \times [0, T]$

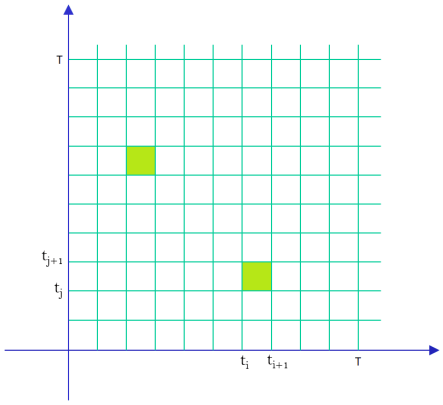

Fig. A.2.: Symmetry of g_2^n

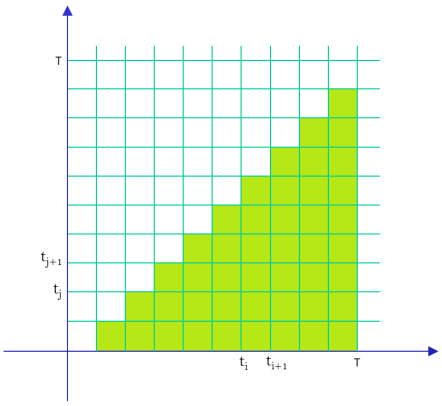

Fig. A.3.: Rectangles participating in S_2^n

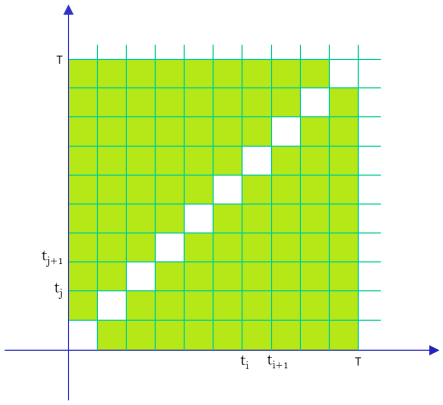

Fig. A.4.: Rectangles in $\bar{S}_2^n$

here that S_2^n is the sum of C_n^2 terms, while $\bar{S}_2^n$ is made out of $V_n^2 = 2!C_n^2$ terms. This is especially apparent when we notice the discretization of $[0, T] \times [0, T]$ is composed of n^2 rectangles, while the diagonal is composed of n rectangles, hence $\bar{S}_2^n$ is the sum of $n^2 - n = V_n^2$ terms. The ratio of the collective mass of the rectangles in the diagonal with respect to the entire $[0, T] \times [0, T]$ goes to zero as $n \to \infty$, hence we conclude that

$$\bar{S}_2^n = \sum_{i=0}^{n-1} \sum_{j \neq i} \frac{\partial f(z_i)}{\partial z_i} \frac{\partial f(z_j)}{\partial z_j} \epsilon^2 \to \int_{[0,T]\times[0,T]} g_2(t, u) d(t, u) = \left(\int_0^T \frac{\partial f(z_t)}{\partial z_t} dt\right)^2, \tag{A.8}$$

thus,

$$S_2^n = \frac{1}{2!} \bar{S}_2^n \to \frac{1}{2!} \left(\int_0^T \frac{\partial f(z_t)}{\partial z_t} dt\right)^2. \tag{A.9}$$

Similarly for

$$S_3^n = \sum_{i=0}^{n-1} \sum_{j<i} \sum_{k<j} \frac{\partial f(z_i)}{\partial z_i} \frac{\partial f(z_j)}{\partial z_j} \frac{\partial f(z_k)}{\partial z_k} \epsilon^3,$$

we can define

$$g_3^n(x, y, z) = \frac{\partial f(z_{t_i})}{\partial z_{t_i}} \frac{\partial f(z_{t_j})}{\partial z_{t_j}} \frac{\partial f(z_{t_k})}{\partial z_{t_k}}, \text{ for } x \in [t_i, t_{i+1}], y \in [t_j, t_{j+1}], z \in [t_k, t_{k+1}], \tag{A.10}$$

and

$$g_3(t,u,v) = \frac{\partial f(z_t)}{\partial z_t}\frac{\partial f(z_u)}{\partial z_u}\frac{\partial f(z_v)}{\partial z_v}, \text{ for } t \in [0,T], u \in [0,T], v \in [0,T]. \tag{A.11}$$

In this case:

$$g_3^n(x,y,z) = g_3^n(x,y,z) = g_3^n(x,z,y) = g_3^n(y,x,z) =$$
$$= g_3^n(y,z,x) = g_3^n(z,x,y) = g_3^n(z,y,x), \tag{A.12}$$

where each equality corresponds to one of the $6 = 3!$ permutations of (x,y,z). As before we can expand the domain of S_3^n, by adding the rectangles in the discretization of $[0,T]\times[0,T]\times[0,T]$ such that i might be smaller than j, as well as that j could be smaller than k. We denote this expanded sum as $\bar{S}_3^n$, and by the symmetry of g_3^n, we notice that $S_3^n = \frac{1}{3!}\bar{S}_3^n$. This implies that the rectangles participating in $\bar{S}_3^n$, now cover most of $[0,T]\times[0,T]\times[0,T]$, where the only exceptions are the ones when $i=j$ or $j=k$ (or both). The number of rectangles participating in $\bar{S}_3^n$ is $V_n^3 = 3!C_n^3 = n^3 - 3n^2 + 2n$, and since the number of all rectangles in $[0,T]\times[0,T]\times[0,T]$ is n^3, this implies that $3n^2 - 2n$ rectangles are missing in sum $\bar{S}_3^n$ from the cases when $i=j$ or $j=k$ (or both). The ratio of the collective mass of such rectangles with respect to the entire $[0,T]\times[0,T]\times[0,T]$ goes to zero as $n\to\infty$, hence as before:

$$\bar{S}_3^n = \sum_{i=0}^{n-1}\sum_{j\neq i}\sum_{i\neq k\neq j}\frac{\partial f(z_i)}{\partial z_i}\frac{\partial f(z_j)}{\partial z_j}\frac{\partial f(z_k)}{\partial z_k}\epsilon^3 \to \int_{[0,T]\times[0,T]\times[0,T]} g_3(t,u,v)d(t,u,v) \tag{A.13}$$

$$= (\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt)^3, \tag{A.14}$$

implying

$$S_3^n = \frac{1}{3!}\bar{S}_3^n \to \frac{1}{3!}(\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt)^3. \tag{A.15}$$

In a similar fashion we can prove that $S_k^n \to \frac{1}{k!}(\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt)^k$. Thus we conclude that:

$$\frac{dz(T)}{dz(0)} = \frac{1}{0!}I + \frac{1}{1!}\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt + \frac{1}{2!}(\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt)^2 + \frac{1}{3!}(\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt)^3 + ... \tag{A.16}$$

$$\frac{dz(T)}{dz(0)} = e^{\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt}. \tag{A.17}$$

Unfortunately, this result does not hold when the dimensionality of $\boldsymbol{z}(t)$ is larger than one. Indeed, in this case, $\frac{\partial f(\boldsymbol{z}_{t_i})}{\partial \boldsymbol{z}_{t_i}}$ is a matrix, hence $g_2^n(x,y) = \frac{\partial f(\boldsymbol{z}_{t_i})}{\partial \boldsymbol{z}_{t_i}}\frac{\partial f(\boldsymbol{z}_{t_j})}{\partial \boldsymbol{z}_{t_j}}$ is not necessarily symmetric, as the commutator $[\frac{\partial f(\boldsymbol{z}_{t_i})}{\partial \boldsymbol{z}_{t_i}}, \frac{\partial f(\boldsymbol{z}_{t_j})}{\partial \boldsymbol{z}_{t_j}}]$ is not necessarily zero. For this reason, inspired by the previous result we try a different approach. Indeed, we can see from Equation (A.17) that $\frac{dz(T)}{dz(0)}$ is the solution of the following ODE:

$$\frac{d(\frac{dz(t)}{dz(0)})}{dt} = \frac{\partial f(z(t))}{\partial z(t)} \frac{dz(t)}{dz(0)} \tag{A.18}$$

as the initial condition $\frac{dz(t=0)}{dz(0)} = I$. Hence we wish to prove that $\frac{dz(t)}{dz(0)}$ satisfies the same ODE in higher dimensions as well.

We notice that we can write:

$$\boldsymbol{z}(t) = \boldsymbol{z}(0) + \int_0^t f(\boldsymbol{z}(\tau))d\tau = g(\boldsymbol{z}(0), t), \tag{A.19}$$

therefore,

$$\begin{aligned} \frac{d(\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)})}{dt} = \frac{\partial^2 g(\boldsymbol{z}(0), t)}{\partial t \partial \boldsymbol{z}(0)} = \frac{\partial^2 g(\boldsymbol{z}(0), t)}{\partial \boldsymbol{z}(0) \partial t} = \\ = \frac{d(\frac{dg(\boldsymbol{z}(0), t)}{dt})}{d\boldsymbol{z}(0)} = \frac{df(\boldsymbol{z}(t))}{d\boldsymbol{z}(0)} = \frac{df(\boldsymbol{z}(t))}{d\boldsymbol{z}(t)} \frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}. \end{aligned} \tag{A.20}$$

We pause for a moment, in order to highlight the similarity of expression A.20 and the forward sensitivity:

$$\frac{d(\frac{d\boldsymbol{z}(t)}{d\boldsymbol{\theta}})}{dt} = \frac{df(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{d\boldsymbol{\theta}} = \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} \frac{d\boldsymbol{z}(t)}{d\boldsymbol{\theta}} + \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}. \tag{A.21}$$

Now from Expression A.20, we infer that $\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}$ is the solution of the following linear ODE:

$$\frac{d(\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)})}{dt} = \frac{\partial f(\boldsymbol{z}(t))}{\partial \boldsymbol{z}(t)} \frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}. \tag{A.22}$$

If we write $\boldsymbol{Y}(t) = \frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}$, then the equation above becomes:

$$\frac{d\boldsymbol{Y}(t)}{dt} = \frac{\partial f(\boldsymbol{z}(t))}{\partial \boldsymbol{z}(t)} \boldsymbol{Y}(t). \tag{A.23}$$

The general solution of first-order homogeneous linear ODEs is given in [Magnus, 1954; Blanes, 2009], and in our case can be written as follows:

$$\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)} = e^{\boldsymbol{\Omega}(t)} \frac{d\boldsymbol{z}(0)}{d\boldsymbol{z}(0)} = e^{\boldsymbol{\Omega}(t)}, \text{ for } \boldsymbol{\Omega}(t) = \sum_{k=1}^{\infty} \boldsymbol{\Omega}_k(t), \tag{A.24}$$

where

$$\boldsymbol{\Omega}_1(t) = \int_0^t \boldsymbol{A}(t_1)\, dt_1, \tag{A.25}$$

$$\boldsymbol{\Omega}_2(t) = \frac{1}{2}\int_0^t dt_1 \int_0^{t_1} dt_2\, [\boldsymbol{A}(t_1), \boldsymbol{A}(t_2)], \tag{A.26}$$

$$\boldsymbol{\Omega}_3(t) = \tag{A.27}$$

$$= \frac{1}{6}\int_0^t dt_1 \int_0^{t_1} dt_2 \int_0^{t_2} dt_3 \left([\boldsymbol{A}(t_1), [\boldsymbol{A}(t_2), \boldsymbol{A}(t_3)]] + [\boldsymbol{A}(t_3), [\boldsymbol{A}(t_2), \boldsymbol{A}(t_1)]] \right), \tag{A.28}$$

and so on for $k > 3$, where:

$$\boldsymbol{A}(t) = \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}. \tag{A.29}$$

We notice that if $z(t)$ is one dimensional, then this result agrees with the one in the previous approach. To conclude, we have proven the following theorem:

Theorem A.1.1. *Let $\frac{d\boldsymbol{z}(t)}{dt} = f(\boldsymbol{z}(t), \boldsymbol{\theta}, t)$, where f is continuous in t and Lipschitz continuous in $\boldsymbol{z}(t)$, furthermore let $\boldsymbol{z}(t) = \boldsymbol{z}_t(\boldsymbol{z}(0), t)$ satisfy the conditions of Clairaut's theorem. Then the following holds:*

$$\frac{d(\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)})}{dt} = \frac{\partial f(\boldsymbol{z}(t))}{\partial \boldsymbol{z}(t)} \frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}. \tag{A.30}$$

Proof. Since f is continuous in t and Lipschitz continuous in $\boldsymbol{z}$ then due to Picard–Lindelöf theorem, $\boldsymbol{z}(T)$ exists and is unique. Furthermore, since f is Lipschitz continuous in $\boldsymbol{z}(t)$, then $\frac{\partial f(\boldsymbol{z}_t)}{\partial \boldsymbol{z}_t}$ exists almost everywhere. Based on the previous derivations we reach the desired conclusion. □

A.2 The Continuous Generalization of the Total Derivative Decomposition

In Appendix A1, we assumed that $\boldsymbol{z}(t) = \boldsymbol{z}(0) + \int_0^t f(\boldsymbol{z}(\tau), \boldsymbol{\theta}, \tau) d\tau$, which was the the limit of $\epsilon \to 0$ of the previous iterative definition of $\boldsymbol{z}_{i+1} = \boldsymbol{z}_i + \epsilon f(\boldsymbol{z}_i, \boldsymbol{\theta}, t_i)$. Now, we assume that the set of parameters in f is different at each discrete time, that is $\boldsymbol{z}_{i+1} = \boldsymbol{z}_i + \epsilon f_i(\boldsymbol{z}_i, \boldsymbol{\theta}_i, t_i)$, for independent sets $\boldsymbol{\theta}_i$. First, we notice that

$$\boldsymbol{z}_n = \boldsymbol{z}_n(\boldsymbol{z}(t_0), \boldsymbol{\theta}(t_0), \boldsymbol{\theta}(t_1), ..., \boldsymbol{\theta}(t_{n-1})) = \boldsymbol{z}_n(\boldsymbol{z}(t_1), \boldsymbol{\theta}(t_1), \boldsymbol{\theta}(t_2), ..., \boldsymbol{\theta}(t_{n-1})) =$$

$$...\boldsymbol{z}_n(\boldsymbol{z}(t_{k+1}), \boldsymbol{\theta}(t_{k+1}), \boldsymbol{\theta}(t_{k+2}), ..., \boldsymbol{\theta}(t_{n-1})) = ...$$

$$... = \boldsymbol{z}_n(\boldsymbol{z}(t_{n-1}), \boldsymbol{\theta}(t_{n-1})) = \boldsymbol{z}(t_n = T - \epsilon).$$

This can be seen from Figure A.5. Thus

$$\frac{d\boldsymbol{z}_n}{d\boldsymbol{\theta}(t_k)} = \frac{d\boldsymbol{z}_n(\boldsymbol{z}(t_k+\epsilon), \boldsymbol{\theta}(t_k+\epsilon), \boldsymbol{\theta}(t_{k+2}), ..., \boldsymbol{\theta}(t_n))}{d\boldsymbol{\theta}(t_k)} \tag{A.31}$$

$$\frac{d\boldsymbol{z}_n}{d\boldsymbol{\theta}(t_k)} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)} \frac{d\boldsymbol{z}(t_k+\epsilon)}{d\boldsymbol{\theta}(t_k)} + 0 + 0 + ... + 0 = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)} \frac{\partial \boldsymbol{z}(t_k+\epsilon)}{\partial \boldsymbol{\theta}(t_k)}. \tag{A.32}$$

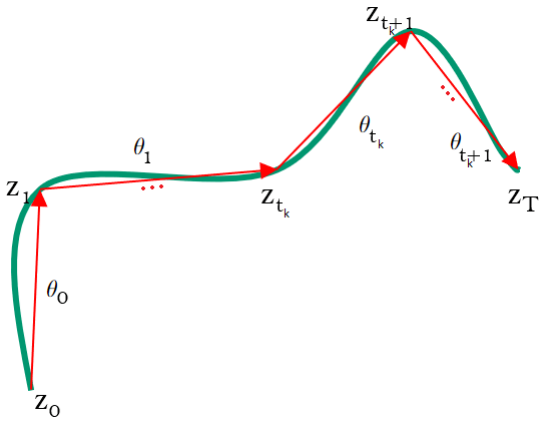

Fig. A.5.: The dependencies in the discretisation of $\frac{\boldsymbol{z}(t)}{t} = f(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t)$. Variable $\boldsymbol{z}_{t_{k+1}}$ completely absorbs the contribution of $\boldsymbol{\theta}_{t_k}$ to $\boldsymbol{z}_T$, hence $\boldsymbol{z}_T$ is a function of $\boldsymbol{z}_{t_{k+1}}$ and the following sets of parameters $\boldsymbol{\theta}_{t_{k+1}}, \boldsymbol{\theta}_{t_{k+2}}, ..., \boldsymbol{\theta}_{t_n}$.

We now focus on analysing $\frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)}$ and $\frac{\partial \boldsymbol{z}(t_k+\epsilon)}{\partial \boldsymbol{\theta}(t_k)}$. First we notice that from:

$$\frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k)} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)} \frac{\partial \boldsymbol{z}(t_k+\epsilon)}{\partial \boldsymbol{z}(t_k)} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)} (I + \epsilon \frac{\partial f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))}{\partial \boldsymbol{z}(t_k)} + O(\epsilon^2)) \tag{A.33}$$

we get

$$\frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k)} - \epsilon \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)} \frac{\partial f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))}{\partial \boldsymbol{z}(t_k)} + O(\epsilon^2). \tag{A.34}$$

Regarding $\frac{\partial \boldsymbol{z}(t_k+\epsilon)}{\partial \boldsymbol{\theta}(t_k)}$ the following holds:

$$\frac{\partial \boldsymbol{z}(t_k+\epsilon)}{\partial \boldsymbol{\theta}(t_k)} = \frac{\partial (\boldsymbol{z}(t_k) + f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))\epsilon + O(\epsilon^2))}{\partial \boldsymbol{\theta}(t_k)} = 0 + \frac{\partial f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))}{\partial \boldsymbol{\theta}(t_k)} \epsilon + O(\epsilon^2). \tag{A.35}$$

After combining both Equation (A.34) and Equation (A.35) in expression (A.32), we deduce that:

$$\frac{d\boldsymbol{z}_n}{d\boldsymbol{\theta}(t_k)} =$$

$$(\frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k)} - \epsilon \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)} \frac{\partial f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))}{\partial \boldsymbol{z}(t_k)} + O(\epsilon^2))(\frac{\partial f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))}{\partial \boldsymbol{\theta}(t_k)} \epsilon + O(\epsilon^2)), \tag{A.36}$$

thus

$$\frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{\theta}(t_k)} = \frac{d\boldsymbol{z}_n}{d\boldsymbol{\theta}(t_k)} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k)} \frac{\partial f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))}{\partial \boldsymbol{\theta}(t_k)} \epsilon + O(\epsilon^2) \tag{A.37}$$

Now assuming that $\boldsymbol{\theta}(t_k) = g(t_k, \boldsymbol{\theta})$, we can write the total derivative of $\boldsymbol{z}_n$ with respect to parameters $\boldsymbol{\theta}$:

$$\frac{d\boldsymbol{z}_n}{d\boldsymbol{\theta}} = \sum_{k=0}^{n} \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{\theta}(t_k)} \frac{d\boldsymbol{\theta}(t_k)}{d\boldsymbol{\theta}} = \sum_{k=0}^{n} \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k)} \frac{\partial f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))}{\partial \boldsymbol{\theta}(t_k)} \frac{d\boldsymbol{\theta}(t_k)}{d\boldsymbol{\theta}} \epsilon + O(\epsilon^2) \tag{A.38}$$

hence taking $\epsilon \to 0$, we conclude that

$$\frac{d\boldsymbol{z}(T)}{d\boldsymbol{\theta}} = \int_0^T \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}(t)} \frac{\partial \boldsymbol{\theta}(t)}{\partial \boldsymbol{\theta}} dt. \tag{A.39}$$

We can see that we are integrating the infinitesimal contributions of parameters $\boldsymbol{\theta}$, at each time t. In case that $\boldsymbol{\theta}(t) = g(t, \boldsymbol{\theta}) = \boldsymbol{\theta}$, then we have

$$\begin{aligned} \frac{d\boldsymbol{z}(T)}{d\boldsymbol{\theta}} &= \int_0^T \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\theta}} dt = \int_0^T \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} I dt = \\ &= \int_0^T \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} dt = -\int_T^0 \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} dt. \end{aligned} \tag{A.40}$$

Theorem A.2.1. *Let $\frac{dz(t)}{dt} = f(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t)$, where f is continuous in t and Lipschitz continuous in $\boldsymbol{z}(t)$ and $\boldsymbol{\theta}(t)$. Then the following holds:*

$$\frac{d\boldsymbol{z}(T)}{d\boldsymbol{\theta}} = \int_0^T \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t)}{\partial \boldsymbol{\theta}(t)} \frac{\partial \boldsymbol{\theta}(t)}{\partial \boldsymbol{\theta}} dt. \tag{A.41}$$

Proof. As before, since f is continuous in t and Lipschitz continuous in z then due to Picard–Lindelöf theorem, $z(T)$ exists and is unique. Furthermore, since f is Lipschitz continuous in $\theta(t)$, then $\frac{\partial f(\boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}(t)}$ exists almost everywhere. Based on the previous derivations we reach the desired conclusion. □

A.3 Generalization of Continuous Backpropagation into Piecewise Continuous Backpropagation

We define $\boldsymbol{z}(t)$ as before, with the only difference being that its derivative is discontinuous at a point (say $\frac{T}{2}$):

$$\boldsymbol{z}(t) = \begin{cases} \boldsymbol{z}(0) + \int_0^t f(\boldsymbol{z}(\tau), \boldsymbol{\theta}(\tau)) d\tau, \ t \in [0, \frac{T}{2}] \\ \boldsymbol{z}(0) + \int_0^{\frac{T}{2}} f(\boldsymbol{z}(\tau), \boldsymbol{\theta}(\tau)) d\tau + \int_{\frac{T}{2}}^t g(\boldsymbol{z}(\tau), \boldsymbol{\phi}(\tau)) d\tau, \ t \in (\frac{T}{2}, T] \end{cases} \tag{A.42}$$

As in [Chen, 2018], we can get

$$\frac{d(\frac{\partial L}{\partial \boldsymbol{z}(t)})}{dt} = \begin{cases} -\frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{z}(t)}, \ t \in [0, \frac{T}{2}] \\ -\frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{\phi}(t))}{\partial \boldsymbol{z}(t)}, \ t \in (\frac{T}{2}, T], \end{cases} \tag{A.43}$$

thus

$$\frac{\partial L}{\partial \boldsymbol{z}(t)} = \begin{cases} \frac{\partial L}{\partial \boldsymbol{z}(T)} - \int_T^t \frac{\partial L}{\partial \boldsymbol{z}(\tau)} \frac{\partial g(\boldsymbol{z}(\tau), \boldsymbol{\phi}(\tau))}{\partial \boldsymbol{z}(\tau)} d\tau, \ t \in (\frac{T}{2}, T] \\ \frac{\partial L}{\partial \boldsymbol{z}(T)} - \int_{\frac{T}{2}}^t \frac{\partial L}{\partial \boldsymbol{z}(\tau)} \frac{\partial f(\boldsymbol{z}(\tau), \boldsymbol{\theta}(\tau))}{\partial \boldsymbol{z}(\tau)} d\tau - \int_T^{\frac{T}{2}} \frac{\partial L}{\partial \boldsymbol{z}(\tau)} \frac{\partial g(\boldsymbol{z}(\tau) \boldsymbol{\phi}(\tau))}{\partial \boldsymbol{z}(\tau)} d\tau, \ t \in (0, \frac{T}{2}]. \end{cases} \tag{A.44}$$

The approach developed in Appendix A2 can be used to generalize continuous backpropagation into piecewise continuous backpropagation. Indeed, identically as before, we have the first order appoximation:

$$\frac{dL_\epsilon}{d\boldsymbol{\theta}} = \sum_{k=0}^{n} \frac{\partial L_\epsilon}{\partial \boldsymbol{z}(t_k)} \frac{\partial f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))}{\partial \boldsymbol{\theta}(t_k)} \frac{\partial \boldsymbol{\theta}(t_k)}{\partial \boldsymbol{\theta}} \epsilon + \sum_{k=0}^{n} \frac{\partial L_\epsilon}{\partial \boldsymbol{z}(t_k)} \frac{\partial g(\boldsymbol{z}(t_k), \boldsymbol{\phi}(t_k))}{\partial \boldsymbol{\phi}(t_k)} \frac{\partial \boldsymbol{\phi}(t_k)}{\partial \boldsymbol{\theta}} \epsilon \tag{A.45}$$

Taking the limit we get:

$$\frac{dL}{d\boldsymbol{\theta}} = \int_0^{\frac{T}{2}} \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}(t)} \frac{\partial \boldsymbol{\theta}(t)}{\partial \boldsymbol{\theta}} dt + \int_{\frac{T}{2}}^{T} \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{\phi}(t))}{\partial \boldsymbol{\theta}(t)} \frac{\partial \boldsymbol{\phi}(t)}{\partial \boldsymbol{\theta}} dt \tag{A.46}$$

and in the same manner we get:

$$\frac{dL}{d\boldsymbol{\phi}} = \int_0^{\frac{T}{2}} \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}(t)} \frac{\partial \boldsymbol{\theta}(t)}{\partial \boldsymbol{\phi}} dt + \int_{\frac{T}{2}}^{T} \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{\phi}(t))}{\partial \boldsymbol{\theta}(t)} \frac{\partial \boldsymbol{\phi}(t)}{\partial \boldsymbol{\phi}} dt, \tag{A.47}$$

If we set $\boldsymbol{\theta}(t) = \boldsymbol{\theta}$ and $\boldsymbol{\phi}(t) = \boldsymbol{\phi}$, then we get:

$$\frac{dL}{d\boldsymbol{\theta}} = \int_0^{\frac{T}{2}} \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} dt, \tag{A.48}$$

and

$$\frac{dL}{d\boldsymbol{\phi}} = \int_{\frac{T}{2}}^{T} \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{\phi})}{\partial \boldsymbol{\phi}} dt. \tag{A.49}$$

A.4 Additional Generated Samples

In Figure A.6a, Figure A.6b and Figure A.6c , are presented additional examples of generated samples of MNIST, CIFAR-10 and CelebA(32×32) by AFFJORD, respectively.

(a) Generated samples of MNIST by AFFJORD.

(b) Generated samples of CIFAR-10 by AFFJORD.

(c) Generated samples of CelebA(32×32.) by AFFJORD

Fig. A.6.: Generated samples by AFFJORD.

In addition in Figure A.7a, Figure A.7b and Figure A.7c, are presented additional examples of generated samples of MNIST, CIFAR-10 and CelebA (32×32) by FFJORD, respectively.

(a) Generated samples of MNIST by FFJORD.

(b) Generated samples of CIFAR-10 by FFJORD.

(c) Generated samples of CelebA(32×32) by FFJORD.

Fig. A.7.: Generated samples by FFJORD.

A.5 Deriving the Instantaneous Change of Variable via the Cable Rule

The trace of a commutator $[\boldsymbol{X}, \boldsymbol{Y}] = \boldsymbol{XY} - \boldsymbol{YX}$ is always zero, hence we prove below that for all terms $\boldsymbol{\Omega}_{k>1}(t)$ given in Equations (A.25) in Appendix A1, we have $tr(\boldsymbol{\Omega}_k(t)) = 0$.

Indeed, since the trace and the integral are interchangeable we have:

$$tr(\boldsymbol{\Omega}_2(t)) = tr\frac{1}{2}\int_0^t dt_1 \int_0^{t_1} dt_2 \,[\boldsymbol{A}(t_1), \boldsymbol{A}(t_2)]$$

$$= \frac{1}{2}\int_0^t dt_1 \int_0^{t_1} dt_2 \, tr[\boldsymbol{A}(t_1), \boldsymbol{A}(t_2)] = 0. \tag{A.50}$$

$$tr(\boldsymbol{\Omega}_3(t)) = \frac{1}{6}\int_0^t dt_1 \int_0^{t_1} dt_2 \int_0^{t_2} dt_3 \Big(tr\big[\boldsymbol{A}(t_1), [\boldsymbol{A}(t_2), \boldsymbol{A}(t_3)]\big] +$$

$$+tr\big[\boldsymbol{A}(t_3), [\boldsymbol{A}(t_2), \boldsymbol{A}(t_1)]\big]\Big) \tag{A.51}$$

$$= \frac{1}{6}\int_0^t dt_1 \int_0^{t_1} (0+0) dt_2 = 0, \tag{A.52}$$

and so on for $k > 3$.

The only exception is the case when $k = 1$:

$$tr(\boldsymbol{\Omega}_1(t)) = tr\int_0^t \boldsymbol{A}(t_1) dt_1 = \int_0^t tr\frac{\partial f(\boldsymbol{z}(t_1), t_1, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t_1)} dt_1. \tag{A.53}$$

Finally, using Jacobi's formula, we conclude that:

$$\log\left|\det\left(\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)}\right)\right| = \log\left|\det\left(e^{\boldsymbol{\Omega}(T)}\right)\right| = \log e^{tr(\boldsymbol{\Omega}(T))} = \int_0^T tr\frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} dt. \tag{A.54}$$

A.6 Simplifications in Section 2.3.1

First we notice that if $z^*(0)$ does not depend on $z(0)$, then

$$\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)} = \Big[I, 0\Big]\, e^{\begin{bmatrix} \int_0^T \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} dt & \int_0^T \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t)} dt \\ \int_0^T \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\phi})}{\partial \boldsymbol{z}(t)} dt & \int_0^T \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t)} dt \end{bmatrix} + \boldsymbol{\Omega}_2(T) + \ldots} \begin{bmatrix} I \\ \frac{d\boldsymbol{z}^*(0)}{d\boldsymbol{z}(0)} \end{bmatrix}, \tag{A.55}$$

becomes

$$\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)} = \Big[I, 0\Big]\, e^{\begin{bmatrix} \int_0^T \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} dt & \int_0^T \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t)} dt \\ \int_0^T \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\phi})}{\partial \boldsymbol{z}(t)} dt & \int_0^T \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t)} dt \end{bmatrix} + \boldsymbol{\Omega}_2(T) + \ldots} \begin{bmatrix} I \\ 0 \end{bmatrix}. \tag{A.56}$$

On the other hand, if the augmented dimensions do not depend on the main dimensions then $\frac{\partial g(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\phi})}{\partial \boldsymbol{z}(t)} = 0$. This implies that

$$\boldsymbol{A}(t) = \begin{bmatrix} \frac{\partial f(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} & \frac{\partial f(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t)} \\ \frac{\partial g(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\phi})}{\partial \boldsymbol{z}(t)} & \frac{\partial g(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t)} \end{bmatrix} \rightarrow \begin{bmatrix} \frac{\partial f(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} & \frac{\partial f(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t)} \\ 0 & \frac{\partial g(\boldsymbol{z}^*(t),\boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t)} \end{bmatrix}. \tag{A.57}$$

Hence,

$$\boldsymbol{\Omega}_1(t) = \int_0^t dt_1 \begin{bmatrix} \frac{\partial f(\boldsymbol{z}(t_1),\boldsymbol{z}^*(t_1),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t_1)} & \frac{\partial f(\boldsymbol{z}(t_1),\boldsymbol{z}^*(t_1),\boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t_1)} \\ 0 & \frac{\partial g(\boldsymbol{z}^*(t_1),\boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t_1)} \end{bmatrix} = \begin{bmatrix} \int_0^t dt_1 \frac{\partial f(\boldsymbol{z}(t_1),\boldsymbol{z}^*(t_1),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t_1)} & \boldsymbol{B}_1(t) \\ 0 & \boldsymbol{D}_1(t) \end{bmatrix} \tag{A.58}$$

$$\boldsymbol{\Omega}_2(t) =$$

$$\int_0^t dt_1 \int_0^{t_1} dt_2 \begin{bmatrix} \frac{\partial f(\boldsymbol{z}(t_2),\boldsymbol{z}^*(t_2),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t_2)} & \frac{\partial f(\boldsymbol{z}(t_2),\boldsymbol{z}^*(t_2),\boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t_2)} \\ 0 & \frac{\partial g(\boldsymbol{z}^*(t_2),\boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t_2)} \end{bmatrix} \begin{bmatrix} \frac{\partial f(\boldsymbol{z}(t_1),\boldsymbol{z}^*(t_1),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t_1)} & \frac{\partial f(\boldsymbol{z}(t_1),\boldsymbol{z}^*(t_1),\boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t_1)} \\ 0 & \frac{\partial g(\boldsymbol{z}^*(t_1),\boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t_1)} \end{bmatrix}$$

$$- \int_0^t dt_1 \int_0^{t_1} dt_2 \begin{bmatrix} \frac{\partial f(\boldsymbol{z}(t_1),\boldsymbol{z}^*(t_1),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t_1)} & \frac{\partial f(\boldsymbol{z}(t_1),\boldsymbol{z}^*(t_1),\boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t_1)} \\ 0 & \frac{\partial g(\boldsymbol{z}^*(t_1),\boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t_1)} \end{bmatrix} \begin{bmatrix} \frac{\partial f(\boldsymbol{z}(t_2),\boldsymbol{z}^*(t_2),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t_2)} & \frac{\partial f(\boldsymbol{z}(t_2),\boldsymbol{z}^*(t_2),\boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t_2)} \\ 0 & \frac{\partial g(\boldsymbol{z}^*(t_2),\boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t_2)} \end{bmatrix}$$

$$= \begin{bmatrix} \int_0^t dt_1 \int_0^{t_1} dt_2 [\frac{\partial f(\boldsymbol{z}(t_1),\boldsymbol{z}^*(t_1),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t_1)}, \frac{\partial f(\boldsymbol{z}(t_2),\boldsymbol{z}^*(t_2),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t_2)}] & \boldsymbol{B}_2(t) \\ 0 & \boldsymbol{D}_2(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Omega}_2^{[z]}(t) & \boldsymbol{B}_2(t) \\ 0 & \boldsymbol{D}_2(t) \end{bmatrix}.$$

In this way we can prove that

$$\boldsymbol{\Omega}(t) = \sum_{k=1}^{\infty} \boldsymbol{\Omega}(t)_k = \begin{bmatrix} \sum_{k=1}^{\infty} \boldsymbol{\Omega}_k^{[z]}(t) & \sum_{k=1}^{\infty} \boldsymbol{B}_k(t) \\ 0 & \sum_{k=1}^{\infty} \boldsymbol{D}_k(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Omega}^{[z]}(t) & \boldsymbol{B}(t) \\ 0 & \boldsymbol{D}(t) \end{bmatrix}.$$

Considering that the exponential of a matrix whose lower left block is zero, will still have a zero lower left block, we have:

$$\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)} = e^{\boldsymbol{\Omega}(t)} = \begin{bmatrix} e^{\boldsymbol{\Omega}^{[z]}(t)} & \bar{\boldsymbol{B}}(t) \\ 0 & \bar{\boldsymbol{D}}(t) \end{bmatrix}.$$

Finally,

$$\left|\det\left(\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)}\right)\right| = \left|\det\left([I,0]\begin{bmatrix} e^{\boldsymbol{\Omega}^{[z]}(t)} & \bar{\boldsymbol{B}}(t) \\ 0 & \bar{\boldsymbol{D}}(t) \end{bmatrix}\begin{bmatrix} I \\ 0 \end{bmatrix}\right)\right| = \left|\det\left(e^{\boldsymbol{\Omega}^{[z]}(t)}\right)\right| =$$

$$e^{\int_0^T tr \frac{\partial f(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} dt + 0 + \ldots 0 + \ldots}. \tag{A.59}$$

A.7 Cable Rule Derived via Equation (2.5

)

Differentiating Equation (2.5) in Appendix A1, we get:

$$\frac{d(\frac{dL}{d\boldsymbol{z}(t)})}{dt} = -\frac{dL}{d\boldsymbol{z}(t)}\frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}. \tag{A.60}$$

Choosing L to be $\boldsymbol{z}(0)$, we have

$$\frac{d(\frac{d\boldsymbol{z}(0)}{d\boldsymbol{z}(t)})}{dt} = -\frac{d\boldsymbol{z}(0)}{d\boldsymbol{z}(t)}\frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}. \tag{A.61}$$

We define $\boldsymbol{A}(t) := \frac{d\boldsymbol{z}(0)}{d\boldsymbol{z}(t)}$, and $\boldsymbol{B}(t) := \boldsymbol{A}(t)^{-1} = \frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}$, so that the equation above becomes

$$\frac{d\boldsymbol{A}(t)}{dt} = -\boldsymbol{A}(t)\frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}, \tag{A.62}$$

thus

$$-\boldsymbol{B}(t)\frac{d\boldsymbol{A}(t)}{dt} = \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}. \tag{A.63}$$

Then from

$$\boldsymbol{B}(t)\boldsymbol{A}(t) = I, \tag{A.64}$$

we have

$$-\boldsymbol{B}(t)\frac{d\boldsymbol{A}(t)}{dt} = \frac{d\boldsymbol{B}(t)}{dt}\boldsymbol{A}(t), \tag{A.65}$$

thus, using this expression in Equation (A.63) we get

$$\frac{d\boldsymbol{B}(t)}{dt}\boldsymbol{A}(t) = \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}. \tag{A.66}$$

Finally, we get the desired result by multiplying both sides from the right with $\boldsymbol{B}(t)$.

A.8 Equivalence Between Continuous Total Derivative Decomposition and the Continuous Backpropagation

In Section 2.3.3. we derived the expression of continuous backpropagation from the continuous total derivative decomposition. However, we can also derive the the formula of the continuous total derivative decomposition (Equation (2.13)) from continuous

backpropagation (Equation (2.14)). Indeed, for a function $f = f(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t)$, where $\boldsymbol{\theta}(t) = g(\boldsymbol{\theta}, t)$, we can see f as $h(\boldsymbol{z}(t), \boldsymbol{\theta}, t) = f(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t)$. Hence,

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \int_0^T \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{dh(\boldsymbol{z}(t), \boldsymbol{\theta}, t)}{d\boldsymbol{\theta}} dt = \int_0^T \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{df(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t)}{d\boldsymbol{\theta}} dt, \tag{A.67}$$

therefore

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \int_0^T \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t)}{\partial \boldsymbol{\theta}(t)} \frac{d\boldsymbol{\theta}(t)}{d\boldsymbol{\theta}} dt. \tag{A.68}$$

Setting $L = \boldsymbol{z}(T)$, gives the desired result.

A.9 Additional Details About the Experiments

As mentioned in the main part of the chapter, we optimized the architecture of FFJORD first and fine-tuned its hyper-parameters.

This architecture remains unchanged in the AFFJORD model for fair comparison, and we only fine-tuned the augmented structure and dimension in addition. Indeed, as it can be seen, the results we report for FFJORD (0.96 MNIST, 3.37 CIFAR) are better than those reported on the original paper (0.99 MNIST, 3.40 CIFAR). The aforementioned improvements were a result of reducing the number of parameters during our tuning process, by reducing the number of CNF blocks from 2 to 1. The total number of parameters for different architectures of FFJORD and AFFJORD can be found on Table A.1. It should be emphasized that in the hypernet architecture, the parameters of AFFJORD are contained inside the parameters in the main architecture of FFJORD, so a bigger model is not being used, simply the flexibility of evolution of the main parameters in time is being increased.

Tab. A.1.: Number of Total Parameters of FFJORD and AFFJORD (Concat and Hypernet Architecture) for MNIST and CIFAR-10. The Multiscale Architecture Is Used in All Cases.

Dataset	Concat		Hypernet	
	FFJORD	AFFJORD	FFJORD	AFFJORD
MNIST	400323	417629	783019	4556169
CIFAR10	679441	820815	1332361	8976115
CelebA (32×32)	679441	820815	1332361	8976115

More generally, our experiments show that augmented architectures generally improve the performance of the standard non-augmented FFJORD counterparts, independently from the baseline architecture used. Furthermore, the farther the performance of FFJORD is from being optimal, the greater are the improvements when using AFFJORD.

Appendix of Chapter 3

B

Contents

B.1 Image Generation and Likelihood Estimation in TPSM and DPSM

Regarding the time-varying Parallel Score Matching (TPSM) approach, we can generate data as in the original framework by iterating the following integration process:

$$\boldsymbol{x}(t_i) = \boldsymbol{x}(t_{i+1}) + \int_{t_i+1}^{t_i} [-\frac{1}{2}b(\tau)(\boldsymbol{x}(\tau) + s_{\boldsymbol{\theta}_i}(\boldsymbol{x}_\tau, \tau))]d\tau, \tag{B.1}$$

and perform likelihood estimation via

$$\log p(\boldsymbol{x}(t_0)) = \log p(\boldsymbol{x}(t_N))$$
$$+\Sigma_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} [-\frac{1}{2}b(\tau)tr\frac{\partial(\boldsymbol{x}(\tau) + s_{\boldsymbol{\theta}_i}(\boldsymbol{x}_\tau, \tau))}{\partial \boldsymbol{x}(\tau)}]d\tau, \tag{B.2}$$

where $s_{\boldsymbol{\theta}_i}(\boldsymbol{x}_t, t) = -\frac{\varepsilon_{\boldsymbol{\theta}_i}(\boldsymbol{x}_t, t)}{\sqrt{1-e^{-\int_0^t b(s)ds}}}$.

Similarly, in the case of Discrete Parallel Score Matching (DPSM), we can easily generate data via back-integrating

$$\boldsymbol{x}(t_i) = \boldsymbol{x}(t_{i+1}) - \epsilon[-\frac{1}{2}b(t_{i+1})(\boldsymbol{x}(t_{i+1}) + s_{\boldsymbol{\theta}_{t_{i+1}}}(\boldsymbol{x}_{t_{i+1}}))]$$

Likewise, we can perform likelihood estimation as follows

$$\log p(\boldsymbol{x}(t_0)) = \log p(\boldsymbol{x}(t_N))$$
$$+\Sigma_{i=1}^{N}[-\epsilon\frac{1}{2}b(t_i)tr\frac{\partial(\boldsymbol{x}(t_i) + s_{\boldsymbol{\theta}_i}(\boldsymbol{x}_{t_i}))}{\partial \boldsymbol{x}(t_i)}].$$

In addition, in both approaches data samples can be generated by integrating the corresponding reverse SDE [Song, 2021].

$$d\boldsymbol{x}(t) = -b(t)[\frac{1}{2}\boldsymbol{x}(t) + \nabla_{\boldsymbol{x}(t)} \log p_t(\boldsymbol{x}(t))]dt + \sqrt{b(t)}d\boldsymbol{w}.$$

B.2 Additional Results

Below we give additional results with regards to the experimental section.

First, we set up a comparison between TPSM2 and SA-DPM, where the number of parameter updates for each network is the same in both approaches. To elaborate,

each network in TPSM2 undergoes 50k parallel parameter updates, and the network in SA-DPM also experiences 50k parameter updates. Given this configuration, the training duration for both approaches is equivalent. However, as displayed in Table B.1, the disparity in their performance is quite significant.

Tab. B.1.: Results comparing the performance of SA-DPM and TPSM2 for the same number of parameter updates. We test the models on CIFAR-10, CelebA, and ImageNet. The results are given in bits/dim (lower is better), the number of parameter updates is given in parentheses, and the training time is given in square brackets.

Method:	CIFAR-10	CelebA 64×64	ImageNet 64×64
SA-DPM	3.30 (50k) [4.5H]	2.38 (50k) [4.5H]	3.76 (150k) [14H]
TPSM2	2.93 (50k) [4.5H]	1.90 (50k) [4.5H]	3.55 (150k) [14H]

Tab. B.2.: Results related to Table 3.4, comparing the number of parameter updates of SA-DPMs and the parallel score matching approaches, namely TPSM and DPSM.

Method:	CIFAR-10	CelebA	ImageNet
SA-DPM	800k	800k	2000k
TPSM0	400k	400k	1000k
TPSM1	100k	100k	250k
TPSM2	50k	50k	150k
DPSM	50k	100k	250k

In addition, below, we provide results as measured by the usual approximation of FID. The number of parameter updates is identical as before (Table B.2). In all cases we generated 50 thousand samples. It is important to notice that while for example the images generated by TPSM2 in Figures B.1 and B.2 are visibly better than those by SA-DPM, the FID score does not reflect this, likely due to it being limited by the several underlying assumptions and biases inflicted by arbitrary choices in its definition [Kynkäänniemi, 2023].

Tab. B.3.: The results given in FID.

Method:	CIFAR-10	CelebA	ImageNet
SA-DPM (3k SDE steps)	12.45	5.34	48.8
TPSM0 (3k SDE steps)	12.88	8.19	51.2
TPSM1 (3k SDE steps)	9.54	7.97	51.9
TPSM2 (3k SDE steps)	12.83	9.23	48.7
SA-DPM (1k SDE steps)	12.47	5.40	51.5
DPSM (1k SDE steps)	**8.89**	**4.88**	**38.06**

Furthermore, we present inferential outcomes in relation to generation and likelihood estimation time, as well as the memory utilization of all the considered approaches. It is

noteworthy that PSM approaches not only abstain from inducing inferential penalties, but also have the potential to expedite generation and likelihood estimation in certain cases, such as DPSM. The batch size for SDE and ODE generation was consistently maintained at 32, whereas for likelihood estimation, it was uniformly reduced to 16. In all instances, the generation of SDEs was executed employing 1000 Numerical Function Evaluations (NFE). In contrast, for ODE generation and likelihood estimation pertaining to SA-DPM and TPSM, a total of 1000 steps employing the RK4 method were taken, equating to 4000 function evaluations. Specifically, in the context of DPSM, a total of 4000 steps were performed using interpolation and integration via the Euler method for both ODE generation and likelihood estimation. The results are provided in Tables B.4, B.5 and B.6.

Tab. B.4.: Inference results on ImageNet. Inference time is given in seconds while memory usage is provided in MB.

Task	Generation SDE	Generation ODE	Likelihood Estimation
SA-DPM	131.8s (2396MB)	674.4s (4260MB)	858.8s (7164MB)
TPSM0	119.8s (2334MB)	635.7s (3951MB)	936.2s (6221MB)
TPSM1	123.6s (2396MB)	626.8s (2800MB)	887.7s (6624MB)
TPSM2	137.1s (2381MB)	567.3s (2392MB)	792.4s (6187MB)
DPSM	124.7s (2709MB)	326.2s (2569MB)	335.2s (2670MB)

Tab. B.5.: Inference results on CelebA. Inference time is given in seconds while memory usage is provided in MB.

Task	Generation SDE	Generation ODE	Likelihood Estimation
SA-DPM	121.2s (2450MB)	662.8s (4156MB)	892.2s (7139MB)
TPSM0	116.3s (2402MB)	599.3s (3704MB)	933.9s (6261MB)
TPSM1	126.9s (2426MB)	661.4s (2789MB)	929.3s (6470MB)
TPSM2	146s (2435MB)	605.5s (2401MB)	837.8s (616.8MB)
DPSM	116.6s (2518MB)	303.1s (2529MB)	318.8s (2650MB)

Tab. B.6.: Inference results on CIFAR-10. Inference time is given in seconds while memory usage is provided in MB.

Task	Generation SDE	Generation ODE	Likelihood Estimation
SA-DPM	38.3s (1890MB)	184.7s (2279MB)	380.2s (3528MB)
TPSM0	38.5s (1780MB)	195.2s (2252MB)	354s (2906MB)
TPSM1	37.9s (1900MB)	198.3s (1989MB)	393.8s (2964MB)
TPSM2	51.1s (1883MB)	185.6s (1895MB)	367s (2920MB)
DPSM	95.2s (1930MB)	171.3s (1959MB)	217.8s (1959MB)

B.3 Additional Generated Samples

Below we provide generated samples from all the models that are present in Table 3.4.

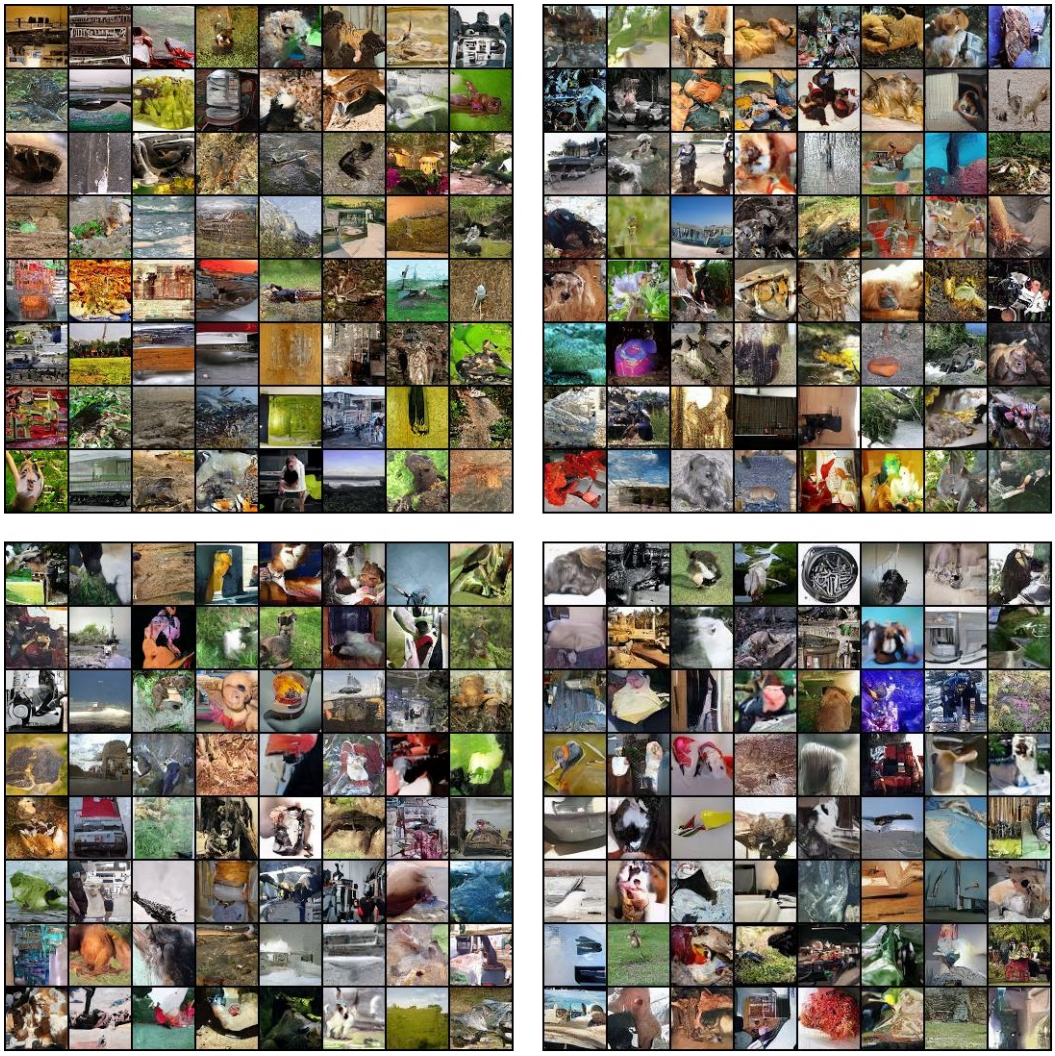

Fig. B.1.: Random non cherry-picked generated ImageNet 64x64 images. Top-Left: SA-DPM, 2000k parameter updates, 1 CNF block, 10k sampling steps via the reverse SDE. Top-Right: TPSM, 250k parameter updates, 10 CNF blocks, 10k sampling steps via the reverse SDE. Bottom-Left: TPSM, 150k parameter updates, 100 CNF blocks, 10k sampling steps via the reverse SDE. Bottom-Right: DPSM, 250k parameter updates, 1000 CNF blocks, 1k sampling steps via the reverse SDE.

Fig. B.2.: Random non cherry-picked generated CIFAR-10 images. Top-Left: SA-DPM, 800k parameter updates, 1 CNF block, 10k sampling steps via the reverse SDE. Top-Right: TPSM, 100k parameter updates, 10 CNF blocks, 10k sampling steps via the reverse SDE. Bottom-Left: TPSM, 50k parameter updates, 100 CNF blocks, 10k sampling steps via the reverse SDE. Bottom-Right: DPSM, 50k parameter updates, 1000 CNF blocks, 1k sampling steps via the reverse SDE.

Fig. B.3.: Random non cherry-picked generated CelebA images. Top-Left: SA-DPM, 800k parameter updates, 1 CNF block, 10k sampling steps via the reverse SDE. Top-Right: TPSM, 100k parameter updates, 10 CNF blocks, 10k sampling steps via the reverse SDE. Bottom-Left: TPSM, 50k parameter updates, 100 CNF blocks, 10k sampling steps via the reverse SDE. Bottom-Right: DPSM, 100k parameter updates, 1000 CNF blocks, 1k sampling steps via the reverse SDE.

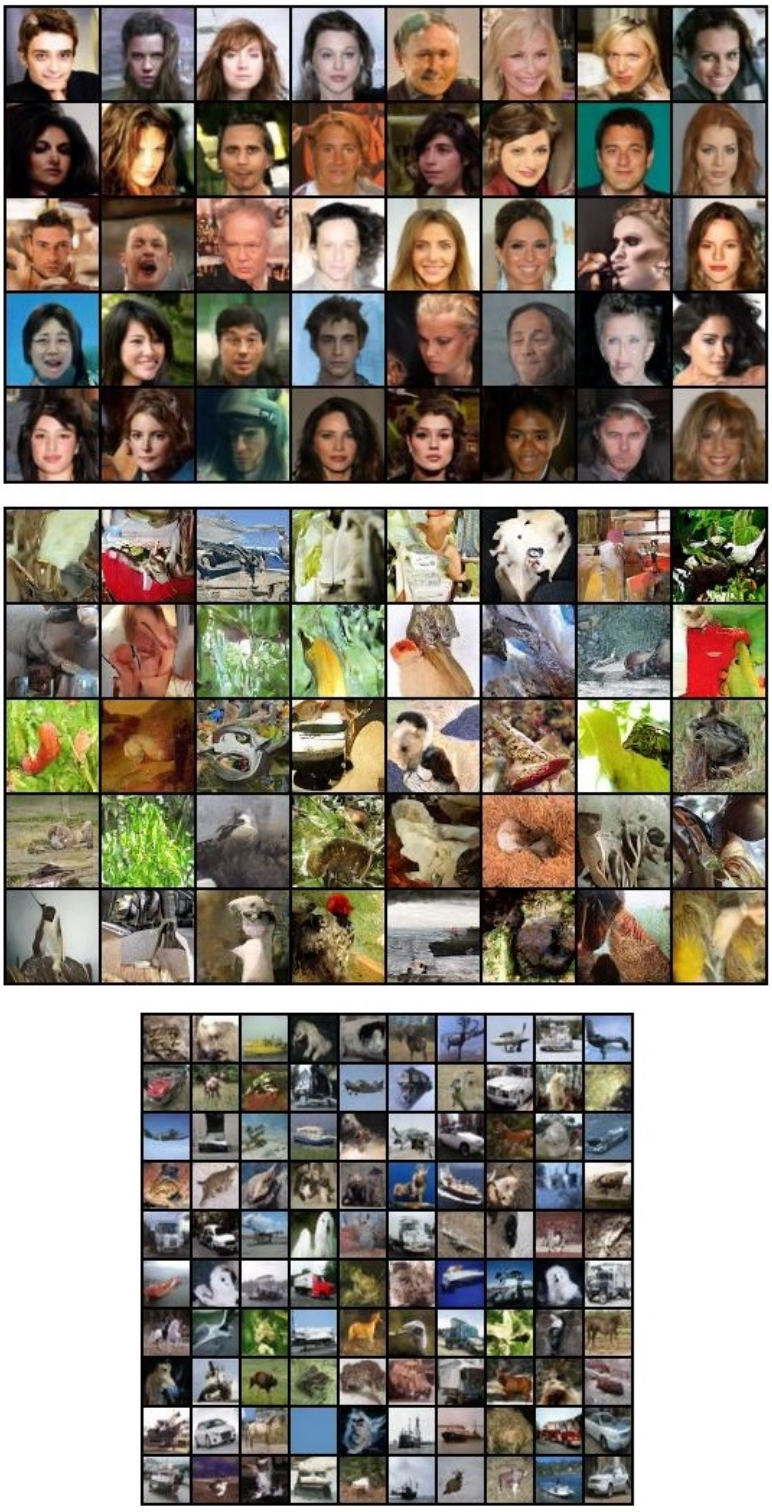

Fig. B.4.: Random non cherry-picked generated CelebA, ImageNet64x64 and Cifar-10 images generated by TPSM0 with 2 CNF blocks and 10000 reverse SDE steps. The number of parameter updates is half that of SA-DPM per block as given in Table B.2.

B.4 The Continuous Evolution of the Score During Diffusion

The SDE process of transforming the data distribution to a normal one is the following:

$$d\boldsymbol{x}(t) = -\frac{1}{2}b(t)\boldsymbol{x}(t)dt + \sqrt{b(t)}d\boldsymbol{w}. \tag{B.3}$$

The equivalent process in terms of the evolution of the distribution is given by the following ODE:

$$d\boldsymbol{x}(t) = -\frac{1}{2}b(t)(\boldsymbol{x}(t) + \nabla_{\boldsymbol{x}(t)} \log p(\boldsymbol{x}(t), t))dt = f(\boldsymbol{x}(t), t)dt. \tag{B.4}$$

The instantaneous change of variable formula states that,

$$\frac{d \log p(\boldsymbol{x}(t), t)}{dt} = -tr\frac{\partial f(\boldsymbol{x}(t), t)}{\partial \boldsymbol{x}(t)}, \tag{B.5}$$

thus by substituting f:

$$\frac{d \log p(\boldsymbol{x}(t), t)}{dt} = \frac{1}{2}b(t)tr\frac{\partial \boldsymbol{x}(t) + \nabla_{\boldsymbol{x}(t)} \log p(\boldsymbol{x}(t), t)}{\partial \boldsymbol{x}(t)} = \frac{1}{2}b(t)D + \frac{1}{2}b(t)tr\frac{\partial \nabla_{\boldsymbol{x}(t)} \log p(\boldsymbol{x}(t), t)}{\partial \boldsymbol{x}(t)}. \tag{B.6}$$

It is easy to notice that,

$$\frac{d\nabla_{\boldsymbol{x}(t)} \log p(\boldsymbol{x}(t), t)}{dt} = \nabla_{\boldsymbol{x}(t)}\frac{d \log p(\boldsymbol{x}(t), t)}{dt} = \frac{1}{2}b(t)\nabla_{\boldsymbol{x}(t)}tr\frac{\partial \nabla_{\boldsymbol{x}(t)} \log p(\boldsymbol{x}(t), t)}{\partial \boldsymbol{x}(t)}. \tag{B.7}$$

If we denote $s(\boldsymbol{x}(t), t) = \nabla_{\boldsymbol{x}(t)} \log p(\boldsymbol{x}(t), t)$, the last expression becomes:

$$\frac{ds(\boldsymbol{x}(t), t)}{dt} = \frac{1}{2}b(t)\nabla_{\boldsymbol{x}(t)}tr\frac{\partial s(\boldsymbol{x}(t), t)}{\partial \boldsymbol{x}(t)}. \tag{B.8}$$

Since,

$$\frac{ds(\boldsymbol{x}(t), t)}{dt} = \frac{\partial s(\boldsymbol{x}(t), t)}{\partial \boldsymbol{x}(t)}\frac{d\boldsymbol{x}(t)}{dt} + \frac{\partial s(\boldsymbol{x}(t), t)}{\partial t} \tag{B.9}$$

using Equation (B.8), we get:

$$\frac{\partial s(\boldsymbol{x}(t), t)}{\partial \boldsymbol{x}(t)}\frac{d\boldsymbol{x}(t)}{dt} + \frac{\partial s(\boldsymbol{x}(t), t)}{\partial t} = \frac{1}{2}b(t)\nabla_{\boldsymbol{x}(t)}tr\frac{\partial s(\boldsymbol{x}(t), t)}{\partial \boldsymbol{x}(t)}, \tag{B.10}$$

and therefore

$$\frac{\partial s(\boldsymbol{x}, t)}{\partial t} = \frac{1}{2}b(t)\nabla_{\boldsymbol{x}}tr\frac{\partial s(\boldsymbol{x}, t)}{\partial \boldsymbol{x}} - \frac{\partial s(\boldsymbol{x}, t)}{\partial \boldsymbol{x}}\frac{d\boldsymbol{x}}{dt}. \tag{B.11}$$

We conclude that the score evolves continuously in time as described by the following PDE:

$$\frac{\partial s(\boldsymbol{x}, t)}{\partial t} = \frac{1}{2}b(t)\nabla_{\boldsymbol{x}}tr\frac{\partial s(\boldsymbol{x}, t)}{\partial \boldsymbol{x}} + \frac{1}{2}b(t)\frac{\partial s(\boldsymbol{x}, t)}{\partial \boldsymbol{x}}(\boldsymbol{x} + s(\boldsymbol{x}, t)). \tag{B.12}$$

This emphasizes the fact that the scores of all the intermediate distributions defined by the FDP are completely determined by the score of the initial (data) distribution.

B.5 DDPM and Flow Matching on 2D toy data

Beyond the investigations delineated in the main chapter's experimental section, we conducted additional experiments involving the utilization of 10 blocks that evenly distribute the diffusion process time interval. These experiments were executed within the DDPM framework, which is consistently utilized throughout the chapter, but also in this special case in the novel Flow-Matching framework.

Tab. B.7.: A comparison of the properties of SA-DPM with TPSM$_{10blocks}$, in the DDPM framework. The results are given in negative log-likelihood (lower is better).

Method:	SA-DPM	TPSM$_{10blocks}$
TY	0.76 (1× **4H**)	0.69 (10× **1H**)
HG	1.11 (1× **1.3H**)	1.05 (10× **0.3H**)

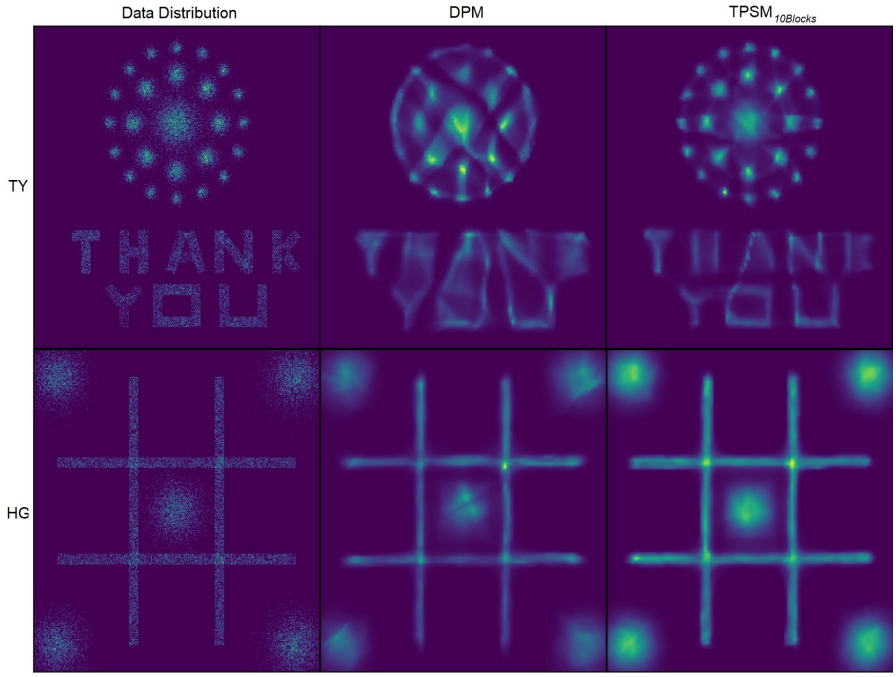

Fig. B.5.: SA-DPM and TPSM with 10 blocks comparison on 2D toy data when the DDPM framework is used.

Within the DDPM framework, the majority of score (probability) evolution occurs when t approaches 0. Consequently, the outcomes of TPSM$_{10Blocks}$ hold potential for substantial enhancement if more recent settings were employed, such as Flow-Matching, as elucidated in [Lipman, 2023], wherein the evolution of the score is more uniformly

distributed over time. This improvement is demonstrated in Table B.8 and Figure B.6. Our observations indicate that the implementation of the Flow-Matching framework bolsters the effectiveness of all approaches. However, the TPSM results exhibit a more pronounced enhancement in the context of the TY data, while in the case of HG the improvements of SA-DPM and TPSM are similar. Thus, even within the confines of more contemporary and efficient frameworks, our parallel score matching strategy continues to deliver the advantageous outcomes postulated in the introduction.

Tab. B.8.: A comparison of the properties of SA-DPM with $\text{TPSM}_{10blocks}$, in the OT Flow-Matching framework. The results are given in negative log-likelihood (lower is better).

Method:	SA-DPM	$\text{TPSM}_{10blocks}$
TY	0.75 (1× **4H**)	0.66 (10× **1H**)
HG	1.10 (1× **1.3H**)	1.04 (10× **0.3H**)

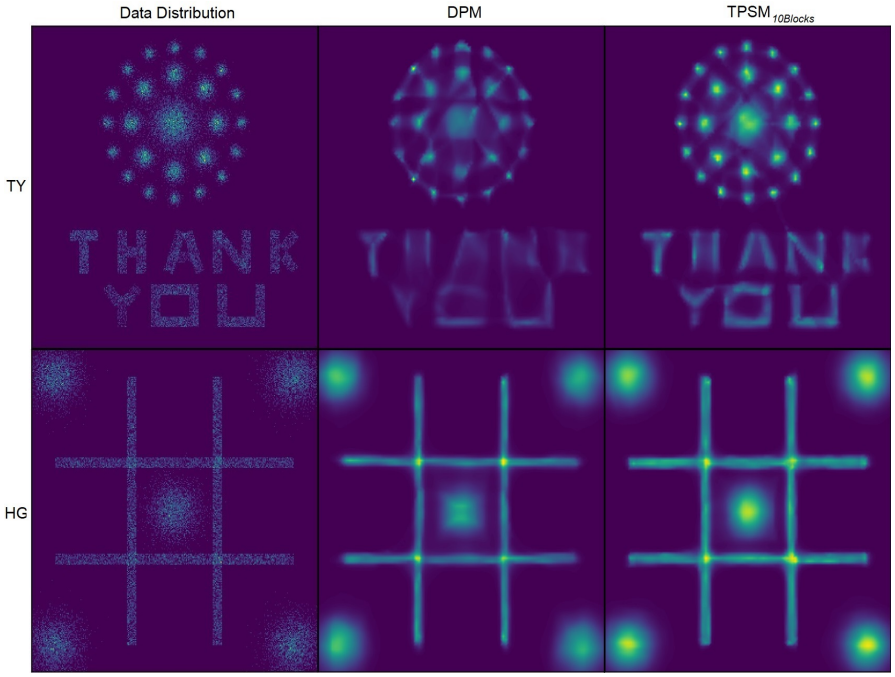

Fig. B.6.: SA-DPM and TPSM with 10 blocks comparison on 2D toy data when the OT Flow-Matching framework is used.

B.6 TPSM-B Visual Results

Owing to the limitations imposed by the margins within the primary part of the chapter, we present here the visual results of TPSM_B, which utilizes 4 neural networks. We notice that even if we train these four networks sequentially, the total training time is the same as in SA-DPM, while the resulting performance shows significant improvements.

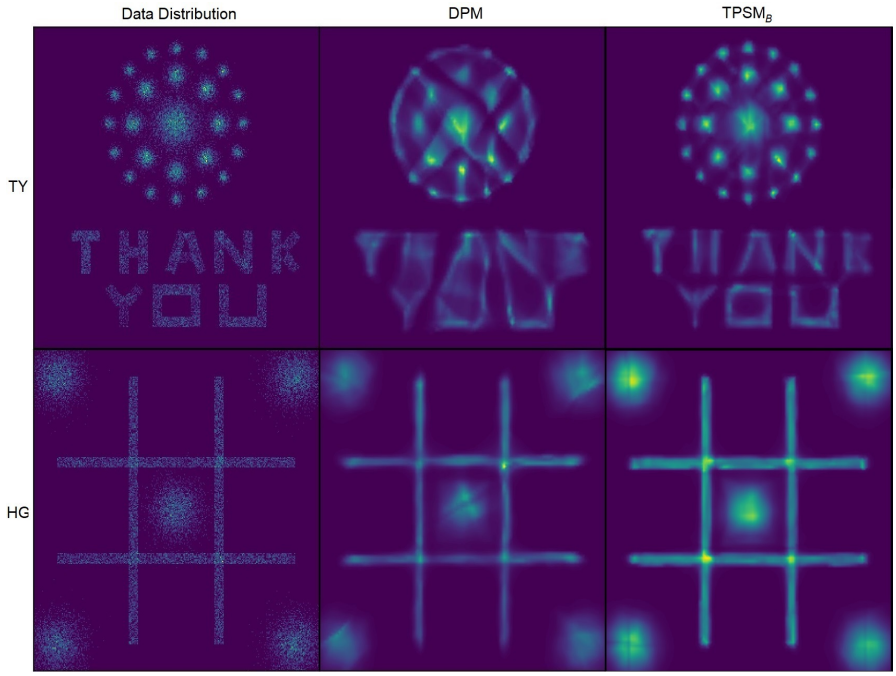

Fig. B.7.: TPSM$_B$ visual results compared with SA-DPM.

B.7 Density Estimation Applied to Time Series Anomaly Detection

From a statistical viewpoint, anomalies are considered as data points residing within the low-probability areas of the feature space within a distribution. Unsupervised anomaly detection, which aims to identify these anomalies, is a key challenge in data analysis, with potential applications in numerous fields. Lately, this domain has attracted significant attention from machine learning researchers, prompted by their ambition to incorporate recent advancements in deep learning. We demonstrate that our PSM strategy, specifically TPSM0, surpasses commonly used methods such as autoencoders. Moreover, we present evidence that even within well-known benchmarks, statistical anomalies do not always match observed anomalies, indicating a discrepancy between the two.

Discrepancy between annotated anomalies and statistical anomalies in the commonly used SWaT dataset

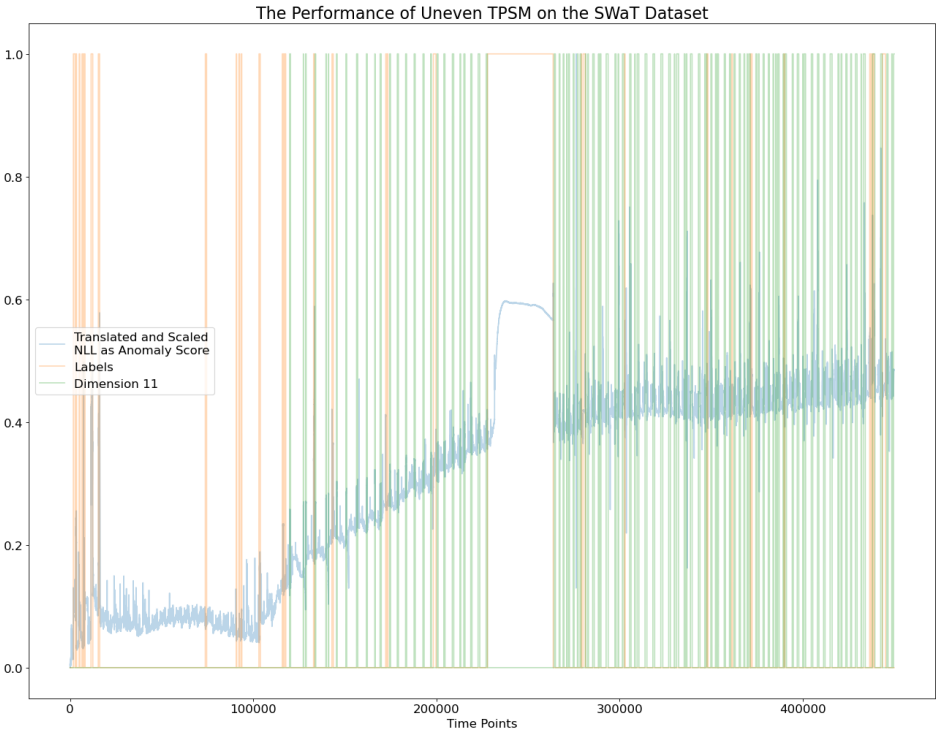

Fig. B.8.: TPSM0 visual results on SWaT.

The Secure Water Treatment (SWaT) dataset [Goh, 2016] is a simplified representation of an actual industrial water treatment plant that produces filtered water. This dataset comprises 11 days of continuous operation, with data from 7 days of normal operations and 4 days of attack scenarios. Each data point in the time series has 51 dimensions. Anomalies are annotated in the test set. Interestingly, the 11th dimension is consistently valued at 0 in all training points but occasionally takes the value 1 in the test set. Nevertheless, the points where the 11th dimension equals 1 are not identified as anomalies despite their statistical significance. We present results showing that an Autoencoder and TPSM0 identify such points as anomalies, even though they are labeled as normal data points. The results are given in Figures B.8 and B.9.

It is important to note that every individual block in TPSM0 underwent a rigorous training process involving 1.5 million parameter updates. This training was conducted with a batch size of 128 to ensure the model's robustness and generalizability. Following this, density estimation was performed using 1000 steps, equivalent to 4000 function evaluations, per point. This intensive procedure ensures the high reliability of our approach.

During this process, we consciously avoided the use of windowing in either of the methodologies, in order not to include temporary information, as our goal was to perform point anomaly detection.

The Autoencoder, in contrast, was subjected to a training regimen of 1200 epochs, ensuring the model learned to discern complex patterns within the data. Intriguingly, both the Autoencoder and TPSM0 managed to identify the points where the 11th dimension is valued at one. Yet, the diffusion model seemed to go a step further, recognizing the entire region containing these values as an anomalous area. This ability of the diffusion model to capture larger anomalous regions potentially positions it as a more sensitive detector for similar tasks in the future.

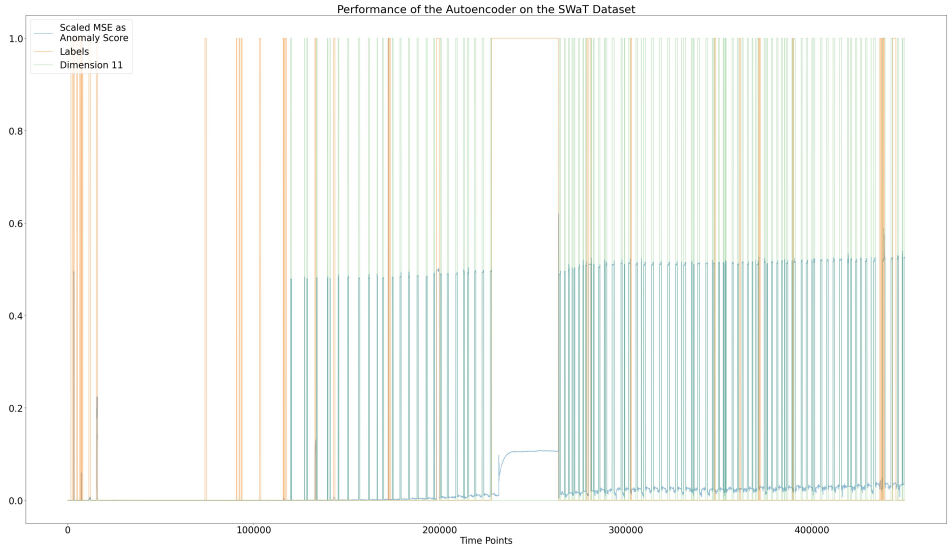

Fig. B.9.: AE visual results on SWaT.

Performance of Diffusion Models on the UCR dataset

In their work, Wu et al. (2021) present a startling assertion that a majority of the datasets used as benchmarks in anomaly detection suffer from one or more of four identified flaws. They argue that these flaws may compromise the reliability of many published comparisons of anomaly detection algorithms. More critically, they suggest that much of the perceived advancements in recent years might be illusory. In their paper, they not only provide evidence to support these claims but also introduce the UCR Time Series Anomaly Archive. They hope that this archive will provide the research community with a benchmark allowing meaningful comparisons between approaches and an accurate measure of overall progress.

We evaluated the performance of the autoencoder (AE) model on the 'UCR Time Series Anomaly Archive' as a baseline in comparison to the TPSM0 method, and our results demonstrate the superior efficacy of TPSM0. We present the percentage of datasets in the UCR archive where we successfully identified the anomalies. Notably, the size of the AE is approximately 3.4 Mb, while TPSM0, comprised of two blocks, maintains a significantly smaller footprint with each block being less than 1.4 Mb. The window size

in both approaches was set to 100. The autoencoder utilizes the MSE as the anomaly score, While TPSM0 uses the negative log-likelihood. The results are presented in Table B.9.

Tab. B.9.: A comparison of the performance of AE with TPSM0, in the UCR time series dataset. The metric used is the percentage of time series in which the anomaly is labeled correctly.

Method	AE	SA-DPM	TPSM0
UCR	51.2%	57.2%	60%

An example where AE misses the anomaly and TPSM0 does not is provided in Figures B.10 and B.11.

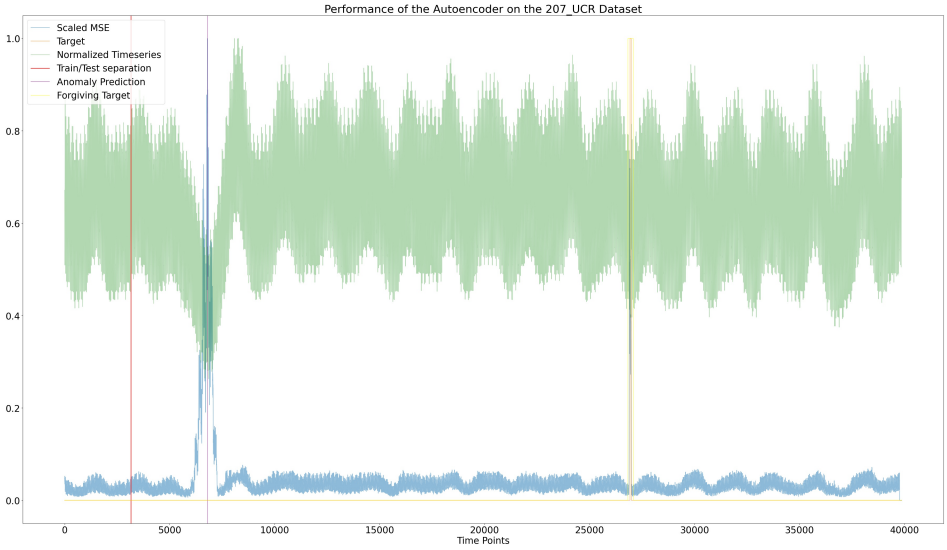

Fig. B.10.: AE misses the anomaly.

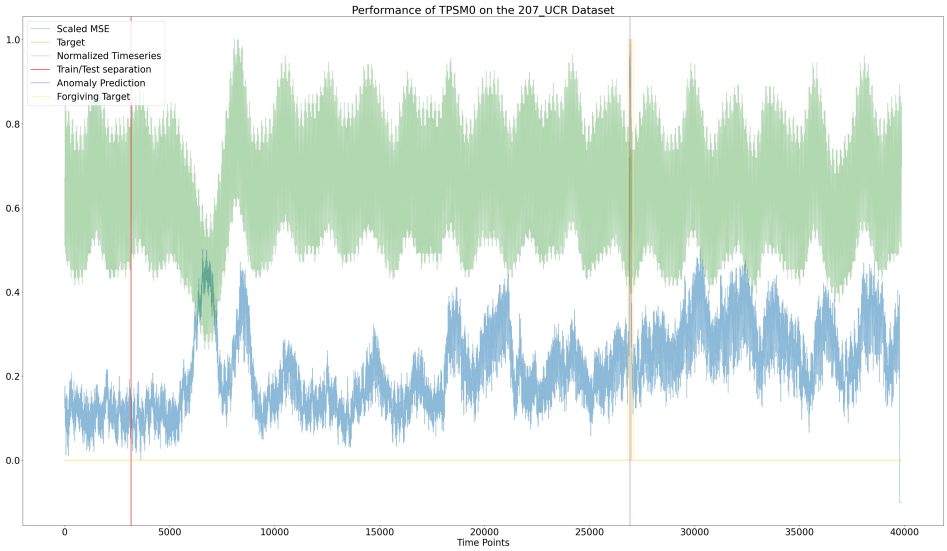

Fig. B.11.: TPSM0 identifies the anomaly correctly.

Appendix of Chapter 4

C

Contents

C.1 Proofs

Proof of Proposition 4.2.9

We notice that if $L(x)$ converges the statement is trivial. However, if it does not then:

$$\lim_{x\to\infty} x^{-\epsilon}L(x) = \lim_{x\to\infty} \frac{L(x)}{x^{\epsilon}} = \lim_{x\to\infty} \frac{e^{c(x)}e^{\int_{x_o}^{x}\frac{u(y)}{y}dy}}{x^{\epsilon}} = \lim_{x\to\infty} \frac{e^{c(x)}e^{\int_{x_o}^{x}\frac{u(y)}{y}dy}}{e^{\epsilon\log(x)}} = \\ = \lim_{x\to\infty} e^{c(x)}e^{\int_{x_o}^{x}\frac{u(y)}{y}dy-\epsilon\log(x)} = \lim_{x\to\infty} e^{c(x)}e^{\log(x)(\frac{\int_{x_o}^{x}\frac{u(y)}{y}dy}{\log(x)}-\epsilon)}. \tag{C.1}$$

Using L'Hopital's rule we get:

$$\lim_{x\to\infty} \frac{\int_{x_o}^{x}\frac{u(y)}{y}}{\log(x)} = \lim_{x\to\infty} \frac{\frac{u(x)}{x}}{\frac{1}{x}} = \lim_{x\to\infty} u(x) = 0, \tag{C.2}$$

therefore

$$\lim_{x\to\infty} e^{\log(x)(\frac{\int_{x_o}^{x}\frac{u(y)}{y}dy}{\log(x)}-\epsilon)} = 0. \tag{C.3}$$

Proof of Lemma 4.4.1

From Theorem 4.2.10, we get that

$$F_1 \in MDA(\xi_1) \iff \bar{F}_1(x) = x^{-\frac{1}{\xi_1}}L_1(x),$$

and

$$F_2 \in MDA(\xi_2) \iff \bar{F}_2(x) = x^{-\frac{1}{\xi_2}}L_2(x),$$

where $L_1(x)$ and $L_2(x)$ are slowly varying functions.
Therefore

$$\lim_{x\to\infty} \frac{\bar{F}_2(x)}{\bar{F}_1(x)} = \lim_{x\to\infty} x^{\frac{1}{\xi_1}-\frac{1}{\xi_2}} \frac{L_2(x)}{L_1(x)} = \lim_{x\to\infty} x^{\alpha}\frac{L_2(x)}{L_1(x)}, \tag{C.4}$$

since

$$\xi_1 > \xi_2 \implies -\frac{1}{\xi_1} > -\frac{1}{\xi_2} \implies \alpha := \frac{1}{\xi_1} - \frac{1}{\xi_2} < 0.$$

On the other hand $L(x) := \frac{L_2(x)}{L_1(x)}$ is defined in a neighborhood of infinity as $L_1(x) \neq 0$, and is also a slowly varying function as

$$\lim_{x\to\infty} \frac{L(ax)}{L(x)} = \lim_{x\to\infty} \frac{\frac{L_2(ax)}{L_1(ax)}}{\frac{L_2(x)}{L_1(x)}} = \lim_{x\to\infty} \frac{\frac{L_2(ax)}{L_2(x)}}{\frac{L_1(ax)}{L_1(x)}} = 1,$$

and since the quotient of positive measurable functions, is positive and measurable. Therefore, using Corollary 1, Equation (C.4) becomes

$$\lim_{x\to\infty}\frac{\bar{F}_2(x)}{\bar{F}_1(x)}=\lim_{x\to\infty}x^{\alpha}\frac{L_2(x)}{L_1(x)}=\lim_{x\to\infty}x^{\alpha}L(x)=0. \tag{C.5}$$

Proof of Lemma 4.4.2

1. If $\xi_1>0$ and $\xi_2=0$ then

$$\lim_{x\to\infty}\frac{\bar{F}_2(x)}{\bar{F}_1(x)}=\lim_{x\to\infty}\frac{c(x)e^{-\int_w^x\frac{g(t)}{a(t)}dt}}{x^{-\frac{1}{\xi}}L(x)}=\lim_{x\to\infty}\frac{c(x)e^{-\log(x)(\frac{\int_w^x\frac{g(t)}{a(t)}dt}{\log(x)}-\frac{1}{\xi})}}{L(x)}, \tag{C.6}$$

using L'Hopital's rule:

$$\lim_{x\to\infty}\frac{\int_w^x\frac{g(t)}{a(t)}dt}{\log(x)}=\lim_{x\to\infty}\frac{\frac{g(x)}{a(x)}}{\frac{1}{x}}=\lim_{x\to\infty}\frac{x}{a(x)}, \tag{C.7}$$

we distinguish two cases:

if $\lim_{x\to\infty}a(x)\neq\infty$ then $\lim_{x\to\infty}\frac{x}{a(x)}=\infty$,

while if $\lim_{x\to\infty}a(x)=\infty$ then using L'Hopital's rule again, we obtain

$$\lim_{x\to\infty}\frac{x}{a(x)}=\lim_{x\to\infty}\frac{1}{a'(x)}=\infty. \tag{C.8}$$

Thus, in both cases

$$=\lim_{x\to\infty}\frac{c(x)e^{-\log(x)(\frac{\int_w^x\frac{g(t)}{a(t)}dt}{\log(x)}-\frac{1}{\xi})}}{L(x)}=\lim_{x\to\infty}\frac{c(x)x^{-(\frac{\int_w^x\frac{g(t)}{a(t)}dt}{\log(x)}-\frac{1}{\xi})}}{L(x)}=0. \tag{C.9}$$

Statements 2. 3. and 4. are trivial. **Proof of Lemma 4.4.3** Since $L(x)$ is positive and measurable (linear combination of finite measurable functions), the only part left to prove is that

$$\lim_{x\to\infty}\frac{L(ax)}{L(x)}=1,\forall a>0.$$

First we prove that

$$\lim_{x\to\infty}\frac{L_1(ax)+L_2(ax)}{L_1(x)+L_2(x)}=1,\forall a>0.$$

Indeed, for each $\epsilon>0$, there exist x_1,x_2 such that for $x>x_1$ we have $|\frac{L_1(ax)}{L_1(x)}-1|<\epsilon$ and for $x>x_2$ we have $|\frac{L_2(ax)}{L_2(x)}-1|<\epsilon$. Hence for $x_0=\max\{x_1,x_2\}$, $x>x_0$ implies

$|L_1(ax) - L_1(x)| < L_1(x)\epsilon$ and $|L_2(ax) - L_2(x)| < L_2(x)\epsilon$ therefore $|L_1(ax) + L_2(ax) - (L_1(x) + L_2(x))| = |L_1(ax) - L_1(x) + L_2(ax) - L_2(x)| \leq |L_1(ax) - L_1(x)| + |L_2(ax) - L_2(x)| < (L_1(x) + L_2(x))\epsilon$ hence $|\frac{L_1(ax)+L_2(ax)}{L_1(x)+L_2(x)} - 1| < \epsilon$.
Now, we notice that for every $a_i > 0$, we get $\lim_{x\to\infty} \frac{a_i L_i(ax)}{a_i L_i(x)} = 1$, and $a_i L_i(x)$ is positive as well as measurable. This implies that $a_1 L_1$ and $a_2 L_2$ are slowly varying functions, and therefore based of the previous result we get

$$\lim_{x\to\infty} \frac{a_1 L_1(ax) + a_2 L_2(ax)}{a_1 L_1(x) + a_2 L_2(x)} = 1, \forall a > 0.$$

Using induction finishes the proof of the Lemma.

Proof of Theorem 4.4.4 Since if $\xi_{\boldsymbol{z}_i} < 0$ then $\exists x_0 > 0$, such that $\forall x > x_0$ we have $F_{\boldsymbol{Z}_i}(x) = 0$, this means that the tail of the distribution is not affected by $F_{\boldsymbol{Z}_i}(x)$. In fact if $\xi_{\max} < 0$ then F will have finite support hence $\xi_F \leq 0$. Furthermore if $\xi_{\max} = 0$ from Lemma 4.4.2 we get that $\xi_F \leq 0$. Therefore for the case $\xi_{\max} > 0$ we only consider the setting where $\xi_i \geq 0$.

$$\bar{F}_u(w) = \frac{1 - F(u+w)}{1 - F(u)} = \frac{\sum_i^n p_i(1 - F_{\boldsymbol{z}_i}(u+w))}{\sum_i^n p_i(1 - F_{\boldsymbol{z}_i}(u))} = \sum_i^n \frac{\bar{F}_{\boldsymbol{z}_i}(u+w)}{\sum_j^n \frac{p_j}{p_i}\bar{F}_{\boldsymbol{z}_j}(u)} \tag{C.10}$$

$$= \sum_i^n \frac{\bar{F}_{\boldsymbol{z}_i}(u+w)}{\bar{F}_{\boldsymbol{z}_i}(u)} \frac{\bar{F}_{\boldsymbol{z}_i}(u)}{\sum_j^n \frac{p_j}{p_i}\bar{F}_{\boldsymbol{z}_j}(u)} = \sum_i^n \frac{\bar{F}_{\boldsymbol{z}_i}(u+w)}{\bar{F}_{\boldsymbol{z}_i}(u)} \frac{1}{\sum_j^n \frac{p_j}{p_i}\frac{\bar{F}_{\boldsymbol{z}_j}(u)}{\bar{F}_{\boldsymbol{z}_i}(u)}}. \tag{C.11}$$

We denote with $i(\max)$ the index corresponding to $\xi_{\max}$ and finish our proof using Pickand's theorem:

$$\lim_{u\to\infty} \sup_{w\in[0,\infty]} |\bar{F}_u(y) - \bar{G}_{\xi_{\max},g(u)}| = \lim_{u\to\infty} \sup_{w\in[0,\infty]} |\sum_i^n \frac{\bar{F}_{\boldsymbol{z}_i}(u+w)}{\bar{F}_{\boldsymbol{z}_i}(u)} \frac{1}{\sum_j^n \frac{p_j}{p_i}\frac{\bar{F}_{\boldsymbol{z}_j}(u)}{\bar{F}_{\boldsymbol{z}_i}(u)}} - \bar{G}_{\xi_{\max},g(u)}| \tag{C.12}$$

$$= \lim_{u\to\infty} \sup_{w\in[0,\infty]} |\sum_i^n \frac{\bar{F}_{\boldsymbol{z}_i}(u+w)}{\bar{F}_{\boldsymbol{z}_i}(u)} \frac{1}{1 + \sum_{j\neq i}^n \frac{p_j}{p_i}\frac{\bar{F}_{\boldsymbol{z}_j}(u)}{\bar{F}_{\boldsymbol{z}_i}(u)}} - \bar{G}_{\xi_{\max},g(u)}| \tag{C.13}$$

$$\begin{aligned} &\leq \lim_{u\to\infty} \sup_{w\in[0,\infty]} |\frac{\bar{F}_{\boldsymbol{z}_{i(\max)}}(u+w)}{\bar{F}_{\boldsymbol{z}_{i(\max)}}(u)} \frac{1}{1 + \sum_{j\neq i(\max)}^n \frac{p_j}{p_{i(\max)}}\frac{\bar{F}_{\boldsymbol{z}_j}(u)}{\bar{F}_{\boldsymbol{z}_{i(\max)}}(u)}} - \bar{G}_{\xi_{\max},g(u)}| \\ &+ \lim_{u\to\infty} \sup_{w\in[0,\infty]} |\sum_{i\neq i(\max)}^n \frac{\bar{F}_{\boldsymbol{z}_i}(u+w)}{\bar{F}_{\boldsymbol{z}_i}(u)} \frac{1}{1 + \sum_{j\neq i}^n \frac{p_j}{p_i}\frac{\bar{F}_{\boldsymbol{z}_j}(u)}{\bar{F}_{\boldsymbol{z}_i}(u)}}| \end{aligned} \tag{C.14}$$

$$
\begin{aligned}
&\leq \lim_{u\to\infty} \sup_{w\in[0,\infty]} |\frac{\bar{F}_{\boldsymbol{z}_{i(\max)}}(u+w)}{\bar{F}_{\boldsymbol{z}_{i(\max)}}(u)} - \bar{G}_{\xi_{\max},g(u)}| \\
&+ \lim_{u\to\infty} \sup_{w\in[0,\infty]} |\frac{1}{1+\sum\limits_{j\neq i(\max)}^{n} \frac{p_j}{p_{i(\max)}} \frac{\bar{F}_{\boldsymbol{z}_j}(u)}{\bar{F}_{\boldsymbol{z}_{i(\max)}}(u)}} - 1||\frac{\bar{F}_{\boldsymbol{z}_{i(\max)}}(u+w)}{\bar{F}_{\boldsymbol{z}_{i(\max)}}(u)}| \\
&+ \lim_{u\to\infty} \sup_{w\in[0,\infty]} \sum_{i\neq i(\max)}^{n} |\frac{\bar{F}_{\boldsymbol{z}_i}(u+w)}{\bar{F}_{\boldsymbol{z}_i}(u)}||\frac{1}{1+\sum\limits_{j\neq i}^{n} \frac{p_j}{p_i} \frac{\bar{F}_{\boldsymbol{z}_j}(u)}{\bar{F}_{\boldsymbol{z}_i}(u)}}|
\end{aligned}
\tag{C.15}
$$

$$
\begin{aligned}
&\leq \lim_{u\to\infty} \sup_{w\in[0,\infty]} |\frac{\bar{F}_{\boldsymbol{z}_{i(\max)}}(u+w)}{\bar{F}_{\boldsymbol{z}_{i(\max)}}(u)} - \bar{G}_{\xi_{\max},g(u)}| \\
&+ \lim_{u\to\infty} |\frac{1}{1+\sum\limits_{j\neq i(\max)}^{n} \frac{p_j}{p_{i(\max)}} \frac{\bar{F}_{\boldsymbol{z}_j}(u)}{\bar{F}_{\boldsymbol{z}_{i(\max)}}(u)}} - 1| \\
&+ \lim_{u\to\infty} \sum_{i\neq i(\max)}^{n} |\frac{1}{1+\sum\limits_{j\neq i}^{n} \frac{p_j}{p_i} \frac{\bar{F}_{\boldsymbol{z}_j}(u)}{\bar{F}_{\boldsymbol{z}_i}(u)}}|.
\end{aligned}
\tag{C.16}
$$

The first expression,

$$
\lim_{u\to\infty} \sup_{w\in[0,\infty]} |\frac{\bar{F}_{\boldsymbol{z}_{i(\max)}}(u+w)}{\bar{F}_{\boldsymbol{z}_{i(\max)}}(u)} - \bar{G}_{\xi_{\max},g(u)}| \tag{C.17}
$$

goes to zero due to Pickands Theorem while the expression,

$$
\lim_{u\to\infty} |\frac{1}{1+\sum\limits_{j\neq i(\max)}^{n} \frac{p_j}{p_{i(\max)}} \frac{\bar{F}_{\boldsymbol{z}_j}(u)}{\bar{F}_{\boldsymbol{z}_{i(\max)}}(u)}} - 1| \tag{C.18}
$$

converges to 0 as well because from Lemma 4.4.1 we have $\lim_{u\to\infty} \frac{\bar{F}_{\boldsymbol{z}_j}(u)}{\bar{F}_{\boldsymbol{z}_{i(\max)}}(u)} = 0$ for every j. Finally the last expression,

$$
\lim_{u\to\infty} \sum_{i\neq i(\max)}^{n} |\frac{1}{1+\sum\limits_{j\neq i}^{n} \frac{p_j}{p_i} \frac{\bar{F}_{\boldsymbol{z}_j}(u)}{\bar{F}_{\boldsymbol{z}_i}(u)}}| \tag{C.19}
$$

equals 0 since in each sum $\sum\limits_{j\neq i}^{n} \frac{p_j}{p_i} \frac{\bar{F}_{\boldsymbol{z}_j}(u)}{\bar{F}_{\boldsymbol{z}_i}(u)}$, there exists an index j such that $\bar{F}_{\boldsymbol{z}_j}(u) = \bar{F}_{\boldsymbol{z}_{i(\max)}}(u)$, implying that $\sum\limits_{j\neq i}^{n} \frac{p_j}{p_i} \frac{\bar{F}_{\boldsymbol{z}_j}(u)}{\bar{F}_{\boldsymbol{z}_i}(u)} \to \infty$.
In the derivation above we assumed that the $F_{\boldsymbol{z}_{i(\max)}}$ which corresponds to $\xi_{\max}$ is unique. In the case that this is not true we notice that for F_1 and F_2 which share the same corresponding parameter $\xi > 0$ we have

$$
p_1 F_1(x) + p_2 F_2(x) = x^{-\frac{1}{\xi}}(p_1 L_1(x) + p_2 L_2(x)) = x^{-\frac{1}{\xi}} L(x), \tag{C.20}
$$

and since $L(x) > 0$, from Lemma 4.4.3 we have that $L(x)$ is slowly varying, therefore $p_1F_1(x) + p_2F_2(x) \in MDA(\xi)$.

Proof of Proposition 4.4.6 First, we fix $\delta > 0$. We can find a $x(\gamma, \delta) > 0$, such that for $x > x(\gamma, \delta)$, we can bound $x^{-\delta}L_z(x) < \gamma$ for all $z \in A$ simultaneously. This implies that $f_Z(z)x^{-\delta}L_z(x)$ is bounded by $f_z(z)\gamma$. Since $\int_z f_z(z)\gamma dz = \gamma < \infty$, by dominated convergence we get

$$\lim_{x\to\infty} x^{-\delta}\int_A f_Z(z)L_z(x)dz = \lim_{x\to\infty}\int_A f_Z(z)x^{-\delta}L_z(x)dz = \int_A \lim_{x\to\infty} f_Z(z)x^{-\delta}L_z(x)dz = 0. \tag{C.21}$$

Proof of Theorem 4.4.7

We will first assume that $\xi_F > 0$.
Since $\bar{F}(x) = x^{-\frac{1}{\xi_F}}L_F(x)$, for every $\epsilon > 0$:

$$\frac{\bar{F}(x)}{x^{-\frac{1}{\xi_{lo}-\epsilon}}} = \frac{x^{-\frac{1}{\xi_F}}L_F(x)}{x^{-\frac{1}{\xi_{lo}-\epsilon}}} = \frac{\int_A f_{\boldsymbol{Z}}(\boldsymbol{z})x^{-\frac{1}{\xi_{\boldsymbol{z}}}}L_{\boldsymbol{z}}(x)d\boldsymbol{z}}{x^{-\frac{1}{\xi_{lo}-\epsilon}}} = \int_A f_{\boldsymbol{Z}}(\boldsymbol{z})x^{-\frac{1}{\xi_{\boldsymbol{z}}}+\frac{1}{\xi_{lo}-\epsilon}}L_{\boldsymbol{z}}(x)d\boldsymbol{z} = \int_A f_{\boldsymbol{Z}}(\boldsymbol{z})x^{\alpha(\boldsymbol{z})}L_{\boldsymbol{z}}(x)d\boldsymbol{z}. \tag{C.22}$$

We notice that $\xi_{\boldsymbol{z}} \geq \xi_{lo} > \xi_{lo}-\epsilon \implies -\frac{1}{\xi_{\boldsymbol{z}}} \geq -\frac{1}{\xi_{lo}} > -\frac{1}{\xi_{lo}-\epsilon}$ hence $\alpha(\boldsymbol{z}) = -\frac{1}{\xi_{\boldsymbol{z}}}+\frac{1}{\xi_{lo}-\epsilon} > 0$.
Considering that

$$\lim_{x\to\infty}\frac{\bar{F}(x)}{x^{-\frac{1}{\xi_{lo}-\epsilon}}} = \lim_{x\to\infty}\int_A f_{\boldsymbol{Z}}(\boldsymbol{z})x^{\alpha(\boldsymbol{z})}L_{\boldsymbol{z}}(x)dz, \tag{C.23}$$

by using Fatou's lemma:

$$\lim_{x\to\infty}\int_A f_{\boldsymbol{Z}}(\boldsymbol{z})x^{\alpha(\boldsymbol{z})}L_{\boldsymbol{z}}(x)d\boldsymbol{z} \geq \int_A \lim_{x\to\infty} f_{\boldsymbol{Z}}(\boldsymbol{z})x^{\alpha(\boldsymbol{z})}L_{\boldsymbol{z}}(x)d\boldsymbol{z} = \infty, \tag{C.24}$$

we get

$$\lim_{x\to\infty}\frac{x^{-\frac{1}{\xi_{lo}-\epsilon}}}{\bar{F}(x)} = 0, \tag{C.25}$$

implying

$$\lim_{x\to\infty}\frac{x^{-\frac{1}{\xi_{lo}-\epsilon}}}{x^{-\frac{1}{\xi_F}}L_F(x)} = \lim_{x\to\infty}\frac{x^{-\frac{1}{\xi_{lo}-\epsilon}+\frac{1}{\xi_F}}}{L_F(x)} = 0, \tag{C.26}$$

therefore

$$\xi_{lo} - \epsilon < \xi_F, \forall\epsilon > 0 \text{ thus } \xi_{lo} \leq \xi_F. \tag{C.27}$$

Now we turn to prove that $\xi_F \leq \xi_{up}$. As before,

$$\frac{\bar{F}(x)}{x^{-\frac{1}{\xi_{up}+\epsilon}}} = \frac{x^{-\frac{1}{\xi_F}} L_F(x)}{x^{-\frac{1}{\xi_{up}+\epsilon}}} = \frac{\int_A f_{\boldsymbol{Z}}(\boldsymbol{z}) x^{-\frac{1}{\xi_{\boldsymbol{z}}}} L_{\boldsymbol{z}}(x) d\boldsymbol{z}}{x^{-\frac{1}{\xi_{up}+\epsilon}}} = \int_A f_{\boldsymbol{Z}}(\boldsymbol{z}) x^{-\frac{1}{\xi_{\boldsymbol{z}}}+\frac{1}{\xi_{up}+\epsilon}} L_{\boldsymbol{z}}(x) d\boldsymbol{z} = \int_A f_{\boldsymbol{Z}}(\boldsymbol{z}) x^{\beta(\boldsymbol{z})} L_{\boldsymbol{z}}(x) d\boldsymbol{z}. \tag{C.28}$$

We notice that $\xi_{\boldsymbol{z}} \leq \xi_{up} < \xi_{up} + \epsilon \implies -\frac{1}{\xi_{\boldsymbol{z}}} \leq -\frac{1}{\xi_{up}} < -\frac{1}{\xi_{up}+\epsilon}$ hence $\beta(\boldsymbol{z}) = -\frac{1}{\xi_{\boldsymbol{z}}} + \frac{1}{\xi_{up}+\epsilon} < -\delta < 0$. This last inequality, combined with the fact that the family $\{L_{\boldsymbol{z}}(x) | x \in \mathbb{R}\}$ is γ-uniformly sub-polynomial, implies that

$$f_{\boldsymbol{Z}}(\boldsymbol{z}) x^{\beta(\boldsymbol{z})} L_{\boldsymbol{z}}(x) \leq f_{\boldsymbol{Z}}(\boldsymbol{z}) x^{-\delta} L_{\boldsymbol{z}}(x) \leq f_{\boldsymbol{Z}}(\boldsymbol{z}) \gamma, \tag{C.29}$$

for some $\gamma > 0$. Since $\int_{\boldsymbol{z}} f_{\boldsymbol{Z}}(\boldsymbol{z}) \gamma d\boldsymbol{z} = \gamma < \infty$, by dominated convergence

$$\lim_{x\to\infty} \frac{\bar{F}(x)}{x^{-\frac{1}{\xi_{up}+\epsilon}}} = \lim_{x\to\infty} \int_A f_{\boldsymbol{Z}}(\boldsymbol{z}) x^{\beta(\boldsymbol{z})} L_{\boldsymbol{z}}(x) d\boldsymbol{z} \tag{C.30}$$

$$\lim_{x\to\infty} \int_A f_{\boldsymbol{Z}}(\boldsymbol{z}) x^{\beta(\boldsymbol{z})} L_{\boldsymbol{z}}(x) d\boldsymbol{z} = \int_A \lim_{x\to\infty} f_{\boldsymbol{z}}(\boldsymbol{z}) x^{\beta(\boldsymbol{z})} L_{\boldsymbol{z}}(x) d\boldsymbol{z} = 0, \tag{C.31}$$

meaning

$$\lim_{x\to\infty} \frac{\bar{F}(x)}{x^{-\frac{1}{\xi_{up}+\epsilon}}} = 0, \tag{C.32}$$

which implies

$$\lim_{x\to\infty} \frac{x^{-\frac{1}{\xi_F}} L_F(x)}{x^{-\frac{1}{\xi_{up}+\epsilon}}} = \lim_{x\to\infty} x^{\frac{1}{\xi_{up}+\epsilon}-\frac{1}{\xi_F}} L_F(x) = 0, \tag{C.33}$$

therefore we get

$$\xi_{up} + \epsilon > \xi_F, \forall \epsilon > 0 \text{ hence } \xi_F \leq \xi_{up}. \tag{C.34}$$

Now we prove that indeed $\xi_F > 0$. It is simple to show that ξ_F cannot be negative. Indeed, if ξ_F is negative, it means that F has finite support which is not possible as for each fixed x, we have $F_{\boldsymbol{z}}(x) > 0, \forall \boldsymbol{z} \in A$, therefore $\forall x \in \mathbb{R}, F(x) > 0$.
Proving that $\xi_F \neq 0$ is slightly less trivial. For every distribution $G_0 \in MDA(0)$ and for $\epsilon < \xi_{lo}$

$$\frac{\bar{F}(x)}{\bar{G}_0(x)} = \frac{\bar{F}(x)}{x^{-\frac{1}{\epsilon}}} \frac{x^{-\frac{1}{\epsilon}}}{\bar{G}_0(x)} = \frac{\int_A f_{\boldsymbol{Z}}(\boldsymbol{z}) x^{-\frac{1}{\xi_{\boldsymbol{z}}}} L_{\boldsymbol{z}}(x) d\boldsymbol{z}}{x^{-\frac{1}{\epsilon}}} \frac{x^{-\frac{1}{\epsilon}}}{\bar{G}_0(x)}. \tag{C.35}$$

As before we can prove that the first fraction $\frac{\bar{F}(x)}{x^{-\frac{1}{\epsilon}}} \to \infty$. The expression $\frac{x^{-\frac{1}{\epsilon}}}{\bar{G}_0(x)}$ goes to ∞ as well due to Lemma 4.4.2, thus

$$\lim_{x\to\infty} \frac{\bar{F}(x)}{\bar{G}_0(x)} = \infty. \tag{C.36}$$

If ξ_F was 0, then for some $G_0 \in MDA(0)$ we would have

$$\lim_{x\to\infty} \frac{\bar{F}(x)}{\bar{G}_0(x)} = \lim_{x\to\infty} 1 = 1, \tag{C.37}$$

hence $\xi_F \neq 0$.

Finally we prove that, if $\xi_{\boldsymbol{z}}$ is continuous in $\boldsymbol{z}$ and $\xi_{\max}$ exists, then we have $\xi_F = \xi_{\max}$. We will first separate A in two sets A_1, A_2, where $A_1 = \{\boldsymbol{z}|\xi_{\max} - \lambda \leq \xi_{\boldsymbol{z}} \leq \xi_{\max}\}$ and $A_2 = \{\boldsymbol{z}|\xi_{lo} \leq \xi_{\boldsymbol{z}} < \xi_{\max} - \lambda\}$. Since $\xi_{\boldsymbol{z}}$ is continuous, then the pre-image of each of the measurable sets $[\xi_{\max} - \lambda, \xi_{\max}], [\xi_{lo}, \xi_{\max} - \lambda)$ will be measurable. In addition, since $[\xi_{\max} - \lambda, \xi_{\max}]$ and $[\xi_{lo}, \xi_{\max} - \lambda)$ contain an open set, then so will A_1 and A_2, implying that $p_i = \mathbb{P}(A_i) > 0$, where $i \in \{1, 2\}$. Thus,

$$\begin{aligned}\bar{F}(x) = \int_A f_{\boldsymbol{Z}}(\boldsymbol{z})\bar{F}_z(x)d\boldsymbol{z} = p_1 \int_{A_1} \frac{f_{\boldsymbol{Z}}(\boldsymbol{z})}{p_1}\bar{F}_{\boldsymbol{z}}(x)d\boldsymbol{z} + p_2 \int_{A_2} \frac{f_{\boldsymbol{Z}}(\boldsymbol{z})}{p_2}\bar{F}_{\boldsymbol{z}}(x)d\boldsymbol{z} \\ = p_1\bar{F}_1(x) + p_2\bar{F}_2(x).\end{aligned} \tag{C.38}$$

From the first part of the Theorem: $\xi_1 \in [\xi_{\max} - \lambda, \xi_{\max}]$, and $\xi_2 \in [\xi_{lo}, \xi_{\max} - \lambda]$, where $F_i \in MDA(\xi_i),\ i = 1, 2$. On the other hand Theorem 4.4.4 implies that $\xi_F = \xi_1$, therefore $\xi_F \in [\xi_{\max} - \lambda, \xi_{\max}]$ for all $\lambda > 0$. We conclude that $\xi_F = \xi_{\max}$.

Proof of Lemma 4.4.9

We assume that $\xi_F > \epsilon$. Then as in the earlier derivations, due to dominated convergence and Lemmas 4.4.1 and 4.4.2, for any $\delta > 0$, we get:

$$\begin{aligned}\lim_{x\to\infty} \frac{x^{-\frac{1}{\xi_F}} L_F(x)}{x^{-\frac{1}{\epsilon+\delta}}} = \lim_{x\to\infty} \frac{\bar{F}(x)}{x^{-\frac{1}{\epsilon+\delta}}} = \lim_{x\to\infty} \int_A f_{\boldsymbol{Z}}(\boldsymbol{z}) \frac{\bar{F}_{\boldsymbol{z}}(x)}{x^{-\frac{1}{\epsilon+\delta}}} d\boldsymbol{z} \\ = \lim_{x\to\infty} \int_{A^+} f_{\boldsymbol{Z}}(\boldsymbol{z}) \frac{\bar{F}_{\boldsymbol{z}}(x)}{x^{-\frac{1}{\epsilon+\delta}}} d\boldsymbol{z} + \lim_{x\to\infty} \int_{A^-} f_{\boldsymbol{Z}}(\boldsymbol{z}) \frac{\bar{F}_{\boldsymbol{z}}(x)}{x^{-\frac{1}{\epsilon+\delta}}} d\boldsymbol{z} \\ = \int_{A^+} \lim_{x\to\infty} f_{\boldsymbol{Z}}(\boldsymbol{z}) \frac{x^{-\frac{1}{\xi_{\boldsymbol{z}}}}}{x^{-\frac{1}{\epsilon+\delta}}} L_{\boldsymbol{z}}(x) d\boldsymbol{z} + \int_{A^-} \lim_{x\to\infty} f_{\boldsymbol{Z}}(\boldsymbol{z}) \frac{\bar{F}_{\boldsymbol{z}}(x)}{x^{-\frac{1}{\epsilon+\delta}}} d\boldsymbol{z} = 0.\end{aligned} \tag{C.39}$$

therefore $\xi_F < \epsilon + \delta, \forall \delta > 0$, contradicting our assumption $\xi_F > \epsilon$.

Proof of Theorem 4.4.11

The proof is similar to that of the last statement in Theorem 4.4.7. We will first separate A in two sets A_1, A_2, where $A_1 = \{\boldsymbol{z}|\xi_{\max} - \lambda \leq \xi_{\boldsymbol{z}} \leq \xi_{\max}\}$ and $A_2 = \{\boldsymbol{z}|\xi_{\boldsymbol{z}} < \xi_{\max} - \lambda\}$. Since $\xi_{\boldsymbol{z}}$ is continuous, then the pre-image of each of the measurable sets $[\xi_{\max} - \lambda, \xi_{\max}], (-\infty, \xi_{\max} - \lambda)$, will be measurable. In addition, since $[\xi_{\max} - \lambda, \xi_{\max}]$ and

$(-\infty, \xi_{\max} - \lambda)$ contain an open set, then so will A_1 and A_2, implying that $p_i = \mathbb{P}(A_i) > 0$, where $i \in \{1, 2\}$.

$$\begin{aligned}\bar{F}(x) = \int_A f_{\boldsymbol{Z}}(\boldsymbol{z})\bar{F}_z(x)d\boldsymbol{z} = p_1 \int_{A_1} \frac{f_{\boldsymbol{Z}}(\boldsymbol{z})}{p_1}\bar{F}_{\boldsymbol{z}}(x)d\boldsymbol{z} + p_2 \int_{A_2} \frac{f_{\boldsymbol{Z}}(\boldsymbol{z})}{p_2}\bar{F}_{\boldsymbol{z}}(x)d\boldsymbol{z} \\ = p_1\bar{F}_1(x) + p_2\bar{F}_2(x).\end{aligned} \tag{C.40}$$

Based on Theorem 4.4.7 and Lemma 4.4.9: $\xi_1 = \xi_{\max}$, and $\xi_2 \in (-\infty, \xi_{\max} - \lambda]$, where $F_i \in MDA(\xi_i),\ i = 1, 2$. From Theorem 4.4.4, we conclude that $\xi_F = \xi_{\max}$. The last statement in the Theorem, that is, if $\xi_{\max} \leq 0$ then $\xi_F \leq 0$, is simply Corollary 4.4.10.

Proof of Proposition 4.4.13

In the case that $\xi_X > 0$, based on our assumptions there exists $L(x)$ such that

$$\mathbb{P}(X > x) = \bar{F}_X(x) = x^{-\frac{1}{\xi_X}} L_1(x). \tag{C.41}$$

Therefore

$$\bar{F}_Y(x) = \mathbb{P}(Y > x) = \mathbb{P}(X^\alpha > x) = \mathbb{P}(X > x^{\frac{1}{\alpha}}) = (x^{\frac{1}{\alpha}})^{-\frac{1}{\xi_X}} L_1(x^{\frac{1}{\alpha}}) = x^{-\frac{1}{\alpha\xi_X}} L_2(x). \tag{C.42}$$

We conclude that $Y \in MDA(\alpha\xi_X)$. On the other hand if $\xi_X \leq 0$ then $\xi_Y \leq 0$, because if $\xi_Y > 0$, then from the first part we would have $\xi_X = \frac{1}{\alpha}\xi_Y > 0$.

Proof of Proposition 4.4.14 We will first prove the case when $p = 1$. If we fix y and denote with ξ_y^{h-}, ξ_y^{h+} the shape parameters of the left and right tail of $p(\hat{f}_{\boldsymbol{V}}(\boldsymbol{X})|y)$, then assuming that at least one of them is positive, from Proposition 21 we know that the tail shape parameter of $p(|\hat{f}_{\boldsymbol{V}}(\boldsymbol{X})||y)$ is $\xi_y^h = \max\{\xi_y^{h-}, \xi_y^{h+}\}$. We notice now that ξ_y^{h-}, ξ_y^{h+} are the right and left tail shape parameters of $p(-\hat{f}_{\boldsymbol{V}}(\boldsymbol{X})|y)$, therefore they are the right and left tail shape parameters of the distribution $p(y - \hat{f}_{\boldsymbol{V}}(\boldsymbol{X})|y)$. Due to this, if we denote with ξ_y^g the tail shape parameter of $p(|y - \hat{f}_{\boldsymbol{V}}(\boldsymbol{X})||y)$, using Proposition 21 once again we have that $\xi_y^g = \max\{\xi_y^{g+}, \xi_y^{g-}\} = \max\{\xi_y^{h-}, \xi_y^{h+}\} = \xi_y^h$, where ξ_y^{g-}, ξ_y^{g+} are the left and right shape parameters of $p(y - \hat{f}_{\boldsymbol{V}}(\boldsymbol{X})|y)$. If both ξ_y^{h-}, ξ_y^{h+} are non-positive then from Proposition 21, ξ_y^h is non-positive, and furthermore ξ_y^g is non-positive, otherwise we could go in the reverse direction and prove that $\xi_y^g > 0$ implies that either $\xi_y^{g-} = \xi_y^{h+}$ is positive, or that $\xi_y^{g+} = \xi_y^{h-}$ is positive.
Now, we denote by $G_y(s)$ the distribution of $|y - \hat{f}_{\boldsymbol{V}}(\boldsymbol{X})|$ given y, and prove that the family $\{G_y(s)|y \in S\}$ has stable cross-tail variability. For each y we denote with $t_0(y)$ the smallest value after which the sub-polynomial assumption is satisfied by $F_y(t)$. Similarly we define $s_0(y)$ for $G_y(s)$. Since the family $\{F_y(t)|y \in S\}$ has stable cross-tail variability, then each such $t_0(y)$ exists, and furthermore the set $\{t_0(y)|y \in S\}$ is bounded from above.

Since each $s_0(y)$ is only displaced by a magnitude of $|y|$ from $t_0(y)$, and since the set S is bounded, then we can conclude that $\{s_0(y)|y \in S\}$ is bounded from above.
We denote ξ^g, ξ^h the tail shape parameters of $|Y - \hat{f}_{\boldsymbol{V}}(\boldsymbol{X})|$ and $|\hat{f}_{\boldsymbol{V}}(\boldsymbol{X})|$ respectively. Using Theorem 4.4.11 twice we get that if there is at least one $\xi^h_y = \xi^g_y > 0$ then $\xi^h = \max\{\xi^h_y | y \in S\} = \max\{\xi^g_y | y \in S\} = \xi^g > 0$, otherwise $\xi^h \leq 0, \xi^g \leq 0$.
Finally we finish the proof by applying Proposition 22 on $|Y - \hat{f}_{\boldsymbol{V}}(\boldsymbol{X})|$ and $|\hat{f}_{\boldsymbol{V}}(\boldsymbol{X})|$.

C.2 Examples where the regularity conditions do not hold

Below we give examples where the regularity conditions do not hold:

Example 1: Let $f_U(u)$ be a uniform distribution, and $g_u(w)$ an exponential distribution with parameter $\frac{1}{u}$. Clearly, the expectation of $g_u(w)$ at each $u \in (0, 1)$ exists. However for

$$h(w) = \int_0^1 f_U(u) g_u(w) du = \int_0^1 u e^{-uw} du \tag{C.43}$$

the expectation is

$$\int_0^\infty \int_0^1 w f_U(u) g_u(w) du dw = \int_0^1 \int_0^\infty w u e^{-uw} dw du = \int_0^1 \frac{1}{u} du \tag{C.44}$$

In this example, we can see that even though all the distributions $g_u(w)$ have shape parameter 0, the shape parameter of $h(w)$ is bigger or equal to one. This is because the beginning of the exponential behaviour of the tail is delayed indefinitely across the elements of the family, violating the γ-uniform sub-polynomial assumption.

Below we give an example of a family of slowly-varying functions $\{L_z(x)|z \in A\}$, where A is compact and $L_z(x)$ is continuous in x and z, but $\{L_z(x)|z \in A\}$ is not γ-uniformly sub-polynomial. In this case, the non slowly-varying behaviour (non sub-polinomiality) of $L_z(x)$, or in other words, the tail of $F_z(x)$, is postponed indefinitely across the family of $\{F_z(x)|z \in A\}$

Example 2: Let $L_z(x)$, for $z \in [0, 1]$, be defined as below:

$$L_z(x) = \begin{cases} 1 + z x^{4-(z-\frac{1}{x})^2} & \text{for } x \in (1, \frac{1}{z}) \\ 1 + \frac{1}{z^3} & \text{for } x \in (\frac{1}{z}, \infty) \end{cases} \tag{C.45}$$

when $z \neq 0$ and $L_0 = 1$ for $x \in (\frac{1}{z}, \infty)$. For x^{-1} we define $F_z(x) = x^{-1}L_z(x)$, that is:

$$F_z(x) = \begin{cases} x^{-1} + zx^{3-(z-\frac{1}{x})^2} & \text{for } x \in (1, \frac{1}{z}) \\ x^{-1} + \frac{1}{z^3}x^{-1} & \text{for } x \in (\frac{1}{z}, \infty) \end{cases} \tag{C.46}$$

when $z \neq 0$ and $F_0 = x^{-1}$ for $x \in (\frac{1}{z}, \infty)$. One can check that $F_z(x)$ and $L_z(x)$ are continuous in z. On the other hand for a given z, $F_z(\frac{1}{z}) = z + z^{-2}$, meaning that $F_z(\frac{1}{z})$ tends to infinity, when z tends to zero. Therefore $\{L_z(x)|z \in A\}$ is not γ-uniformly sub-polynomial.

C.3 Examples where the regularity conditions hold

Below we give examples where the regularity conditions do hold:

Example 3: Let $\bar{F}_z(x) = x^{-z} = x^{-\frac{1}{\frac{1}{z}=\xi_z}}$ for $z \in (1, \infty)$, and let $\bar{F}(x) = e\int_1^\infty e^{-z}\bar{F}_z(x)dz$. Then $\bar{F}(x) = x^{-1}\frac{1}{1+\ln x} = x^{-1}L(x)$, where $L(x) = \frac{1}{\ln x}$ is slowly varying as both 1 and $\ln x$ are slowly varying.

Example 4: Let $\bar{F}_z(x) = x^{-z}\ln x^z$ for $z \in (1, 2)$, and let $\bar{F}(x) = \int_1^2 \bar{F}_z(x)dz$. Then $\bar{F}(x) = x^{-1} - 2x^{-2} + x^{-1}\frac{1}{\ln x} - x^{-2}\frac{1}{\ln x} = x^{-1}(1 - 2x^{-1} + \frac{1}{\ln x} - x^{-1}\frac{1}{\ln x}) = x^{-1}L(x)$, where $L(x) = 1 - 2x^{-1} + \frac{1}{\ln x} - x^{-1}\frac{1}{\ln x}$ is slowly varying.

C.4 Moment based motivation

In Proposition 4.4.14, we showed that under certain conditions, we could estimate the shape of the tail of the distribution of $W_{\boldsymbol{V}}(\boldsymbol{U})$ without using test labels. This can also be motivated from the moments of $W_{\boldsymbol{V}}(\boldsymbol{U})$. Indeed, conditioning on the test label y we have

$$\mathbb{E}[W^p_{\boldsymbol{V}}(\boldsymbol{U})|Y = y] = E_{\boldsymbol{V}}[(y - \hat{f}_{\boldsymbol{V}}(\boldsymbol{x}))^p|y] \tag{C.47}$$

$$= \sum_{k=0}^{p} \binom{p}{k} y^k(-1)^{p-k}E_{\boldsymbol{V}}[\hat{f}^{p-k}_{\boldsymbol{V}}(\boldsymbol{x})|y] \tag{C.48}$$

We can see that for test label y, if the moment p of $\hat{f}_{\boldsymbol{V}}(\boldsymbol{x})$ given y exists then the moment p of $W_{\boldsymbol{V}}(u)$ given y exists. If each $E_{\boldsymbol{V}}[\hat{f}^j_{\boldsymbol{V}}(\boldsymbol{x})|y]$, $j \in \{1, ..., p\}$ changes continuously with y then $\mathbb{E}[W^p_{\boldsymbol{V}}(\boldsymbol{U})|y]$ is continuous with respect to y. Further assuming that the support of Y is compact, then moment p of $W_{\boldsymbol{V}}(\boldsymbol{U})$, that is, $\mathbb{E}[W^p_{\boldsymbol{V}}(\boldsymbol{U})] = \mathbb{E}_y\mathbb{E}[W^p_{\boldsymbol{V}}(\boldsymbol{U})|Y = y]$ will exist as well.

Under these conditions, if $\hat{f}_{\boldsymbol{V}}(\boldsymbol{x})$ is a non-negative function, then the existence of $\mathbb{E}[\hat{f}^p_{\boldsymbol{V}}(\boldsymbol{x})] = \mathbb{E}_y\mathbb{E}[\hat{f}^p_{\boldsymbol{V}}(\boldsymbol{x})|y]$ guarantees the existence of $\mathbb{E}[\hat{f}^p_{\boldsymbol{V}}(\boldsymbol{x})|y]$ for almost all y, thus it ensures the existence of $\mathbb{E}[W^p_{\boldsymbol{V}}(\boldsymbol{U})]$.

C.5 Reducing the variability of the estimated shape parameters

It is proven in [Dekkers, 1989b], that under certain conditions on k (in particular that $\frac{k(n)}{n} \to 0$ as $n \to \infty$) the Pickands Estimator has an asymptotically Gaussian distribution: $\sqrt{k(n)}(\hat{\xi}^{(P)}_{k,n} - \xi) \xrightarrow{d} \mathcal{N}(0, \sigma^2(\xi))$. This implies that for large n, we roughly have $\hat{\xi}^{(P)}_{k,n} \sim \mathcal{N}(\xi, \frac{\sigma^2(\xi)}{k(n)})$. Minding the size of n, we can split the n samples into m groups such that $n = m\frac{n}{m}$, and such that we still have roughly $\hat{\xi}^{(P)}_{k,\frac{n}{m}} \sim \mathcal{N}(\xi, \frac{\sigma^2(\xi)}{k(\frac{n}{m})})$. Since we can estimate $\hat{\xi}^{(P)}_{k,\frac{n}{m}}$ for each of the m groups we can define the average estimation as $\hat{\xi}^{(P),avg}_{k,\frac{n}{m}} = \frac{1}{m}\sum_{i=1}^{m}\hat{\xi}^{(P),i}_{k,\frac{n}{m}}$. Under the assumption that samples from such groups are independent, we get that $\hat{\xi}^{(P),avg}_{k,\frac{n}{m}} \sim \mathcal{N}(\xi, \frac{\sigma^2(\xi)}{mk(\frac{n}{m})})$. Since $k(n) = o(n)$, we can choose to reduce the variance 'linearly' by keeping $\frac{n}{m}$ constant and increasing m, instead of increasing the sub-linear $k(n)$. This becomes quite apparent if we set $k(n) = \log n$ or $k(n) = \sqrt{n}$. Indeed, for $k(n) = \log n$, the ratio between the variances of the direct approach and our approach is

$$\frac{m\log\frac{n}{m}}{\log n} = \frac{m\log\frac{n}{m}}{\log m + \log\frac{n}{m}} = \frac{mC}{\log m + C} \to \infty \tag{C.49}$$

as $m \to \infty$.

Similarly for $k(n) = \sqrt{n}$,

$$\frac{m\sqrt{\frac{n}{m}}}{\sqrt{n}} = \sqrt{m} \to \infty \tag{C.50}$$

as $m \to \infty$. Here we can see that even if we fix m and then allow each group with size $\frac{n}{m}$ to grow as n increases, the variance is still $\sqrt{m}$ times smaller using our approach.

The asymptotically Gaussian distribution property holds in the case of the DEdH estimator if one knows that $\xi > 0$ (Hill estimator, [Davis, 1984]). Furthermore, both estimators $H^{(1)}_{k,n}$ and $H^{(2)}_{k,n}$ in Definition 4.2.6 jointly possess this property, [Dekkers, 1989a].

C.6 The inadequacy of the direct POT usage on mixture distributions

In this section, we illustrate two cases where cross tail estimation is necessary for proper tail shape estimation. **Uniform Case**

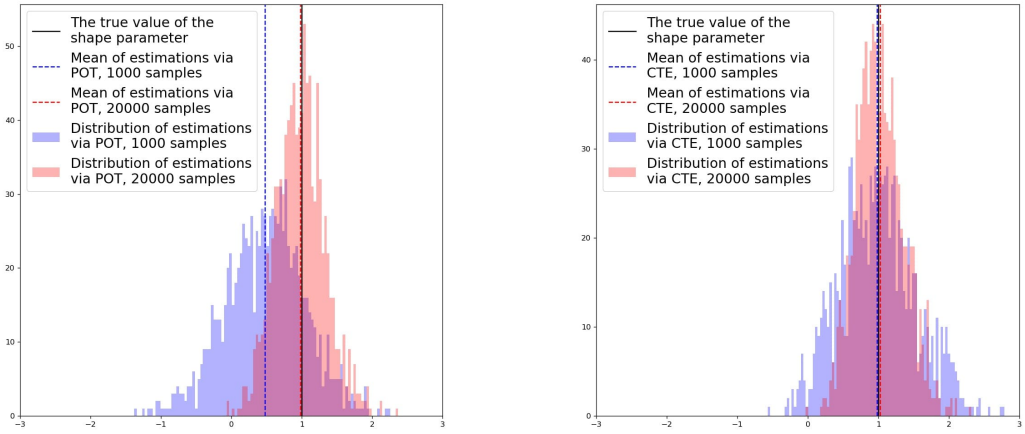

Fig. C.1.: Standard estimation of the shape parameter of the tails by simply applying the Pickands' Estimator, on average, gives poor results on fewer data (left). Cross tail estimation (CTE) gives the correct estimation on average. (right).

In our experimental procedure, we randomly select samples adhering to two distinct power law distributions. Each of these distributions has a unique characteristic shape parameter - one has a shape parameter of 1, while the other possesses a shape parameter of 0.5. For our random sampling process, we afford equal probability, precisely 50%, to both these distributions. This means there is an identical chance of picking a sample from either of these power law distributions, each with their respective shape parameters.

When we examine an experimental set of 10^3 sampled points from each of these distributions, the resulting pattern becomes apparent as shown in Figure C.1 (left). We find that if we amalgamate all the sampled data points from both distributions into a unified array, and subsequently apply Pickands Estimator on this consolidated data set, the process yields a sub-optimal estimation of the distribution tail. The outcome is unsatisfactory as it fails to reveal the accurate shape of the tail, thereby defeating the purpose of the estimation.

However, we discover that there is a noticeable enhancement in the quality of the estimation when we bolster the sample size from the initial 10^3 to a considerably larger

size of $2 * 10^4$. This increase in sample size permits us to retrieve the true shape of the distribution tail.

Using CTE however, we find that a sample size of just 10^3 proves to be adequate in obtaining a satisfactory estimation of the distribution tail. As illustrated in Figure C.1 (right), this method leads to an accurate estimation with a substantially smaller sample size. Therefore, our method introduces an efficient pathway towards achieving accurate estimations with fewer resources, thereby demonstrating its potential superiority over the traditional Pickands Estimator.

Non-Uniform Case

Similarly, in the second experiment, we sample with 20% probability from a distribution with power law tails with shape parameter 1, and with 80% probability probability from a distribution with power law tails with shape parameter 0.5.

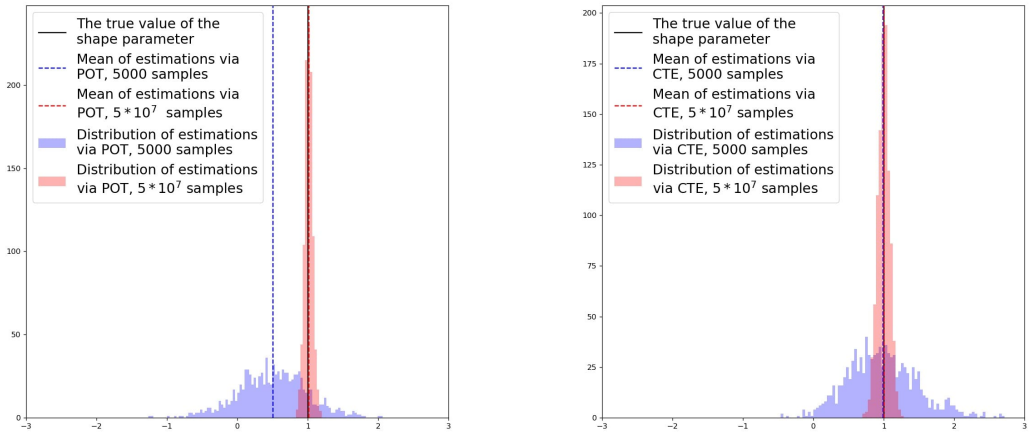

Fig. C.2.: Standard estimation of the shape parameter of the tails by simply applying the Pickands' Estimator, on average, gives poor results on fewer data (left). Cross tail estimation (CTE) gives the correct estimation on average. (right).

When sampling $5 * 10^3$ points from each distribution, Figure C.2, we are not able to properly estimate the tail if we join all the samples together in a common array and then apply the Pickands' Estimator. But, if we increase the sample size from $5 * 10^3$ to $5 * 10^7$, we manage to retrieve the the true tail shape of the mixture. However, using our method, $5 * 10^3$ samples are already sufficient to get a proper estimation.

C.7 Additional details with regards to section 4.5.1

Below we provide Figure C.3 which illustrates how ξ_z evolves depending on the $\xi_{\max}$ which is given as input. The parameter $\xi_{\max}$ takes the following 45 values $\{-4, -4 + 0.1, -4 + 0, 2, ..., 5\}$.

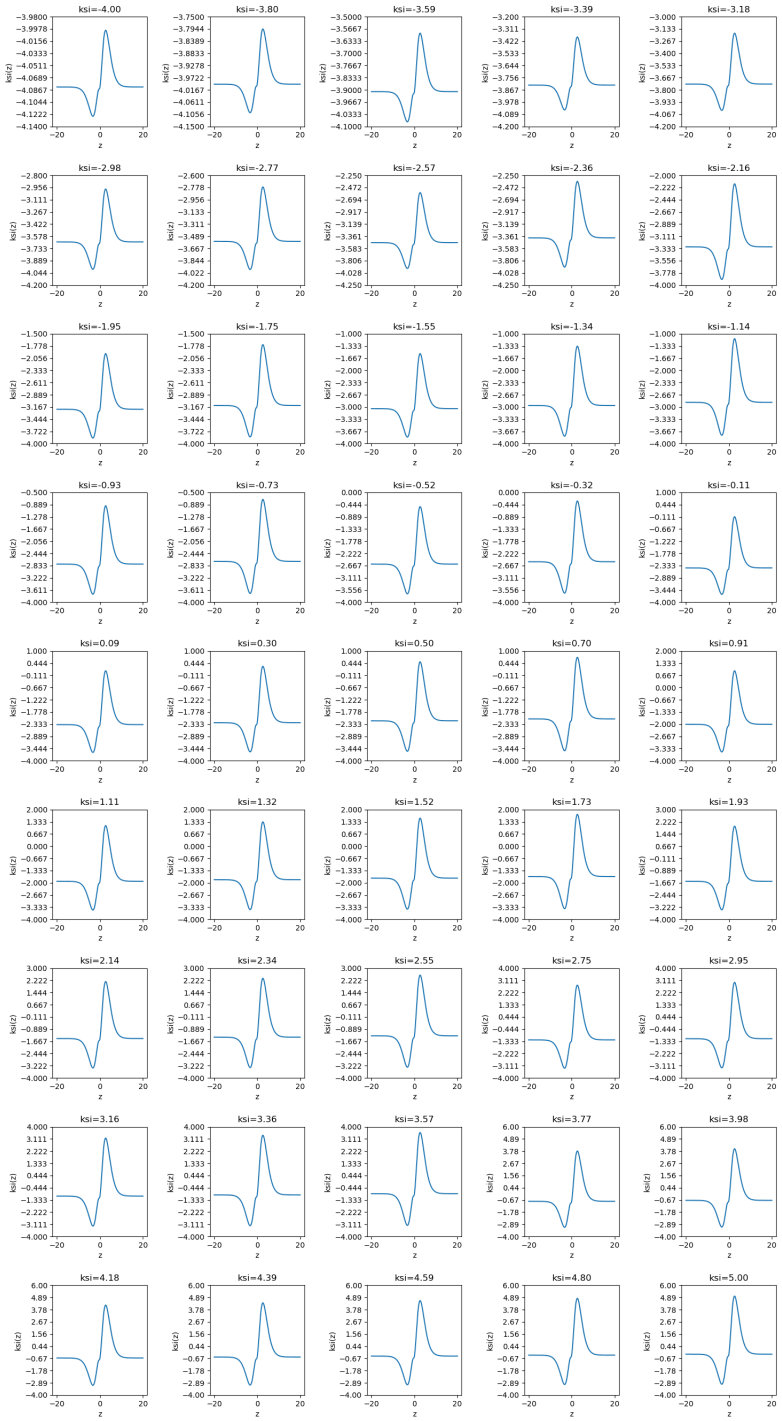

Fig. C.3.: The evolution of ξ_z depending on the value of $\xi_{\max}$.

C.8 Additional details with regards to section 4.5.3

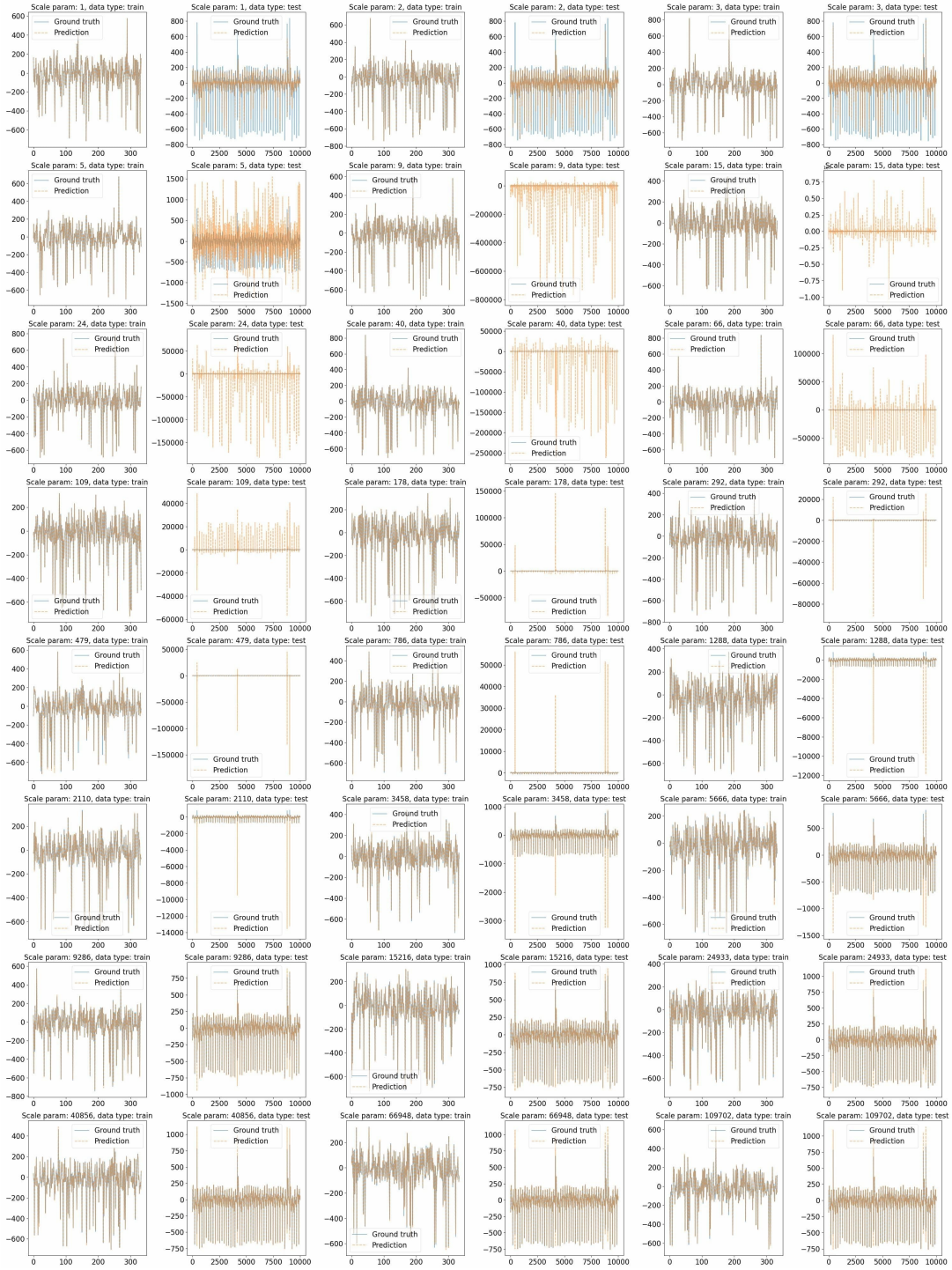

Fig. C.4.: The performance of Gaussian process on train and test data depending on the length scale parameter. First half of the cases.

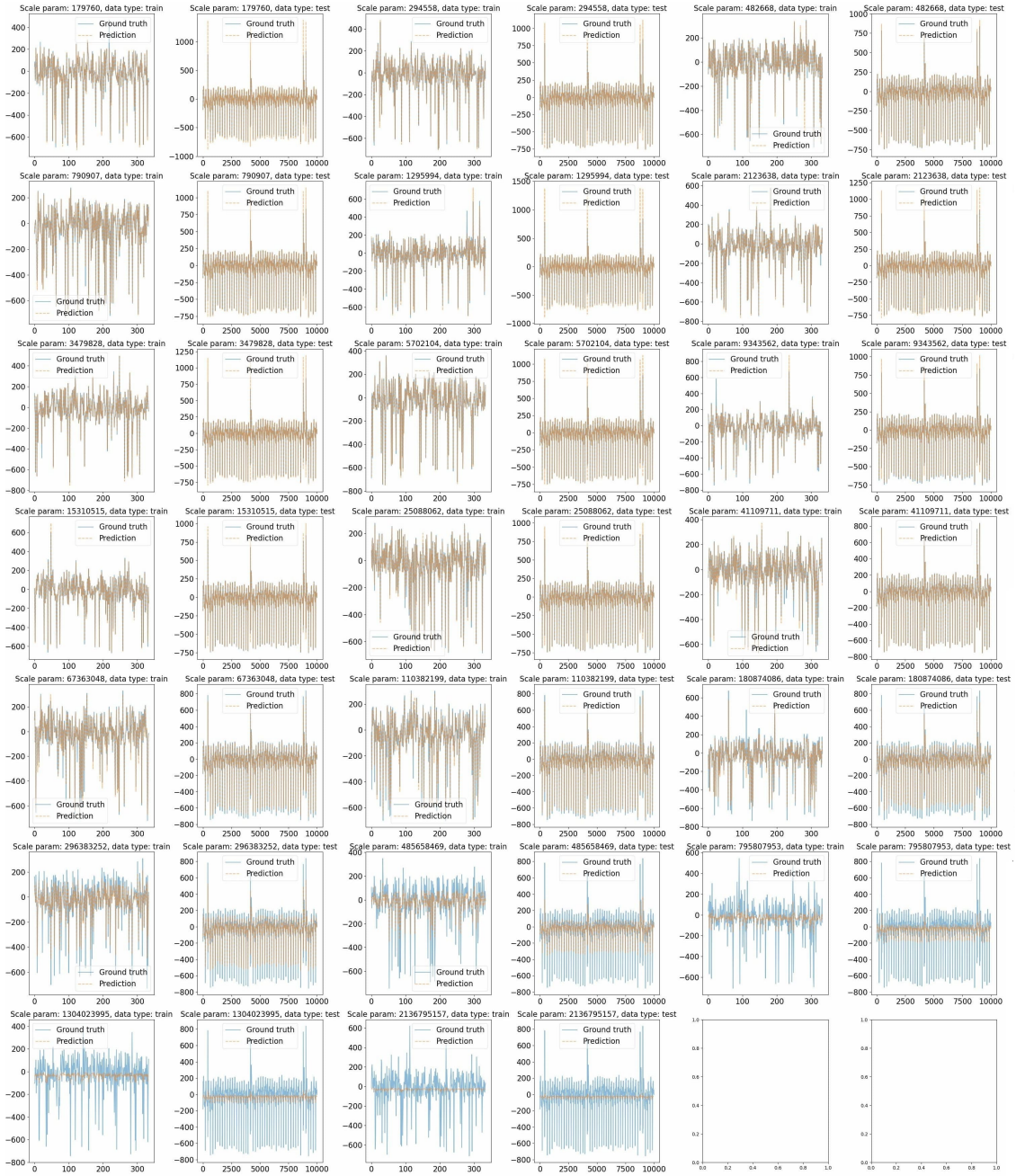

Fig. C.5.: The performance of Gaussian process on train and test data depending on the length scale parameter. Second half of the cases.

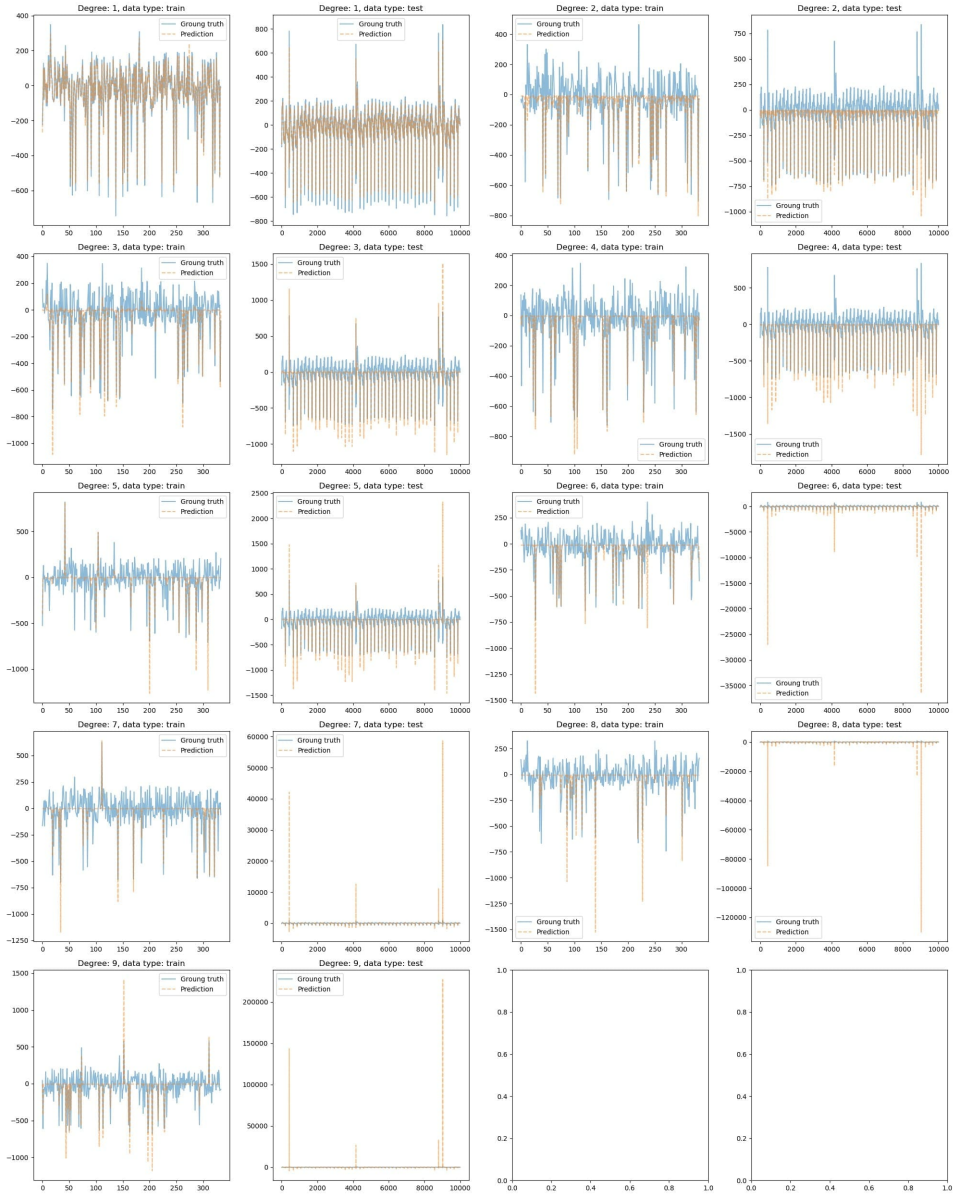

Fig. C.6.: The performance of polynomial kernels on train and test data depending on the degree.

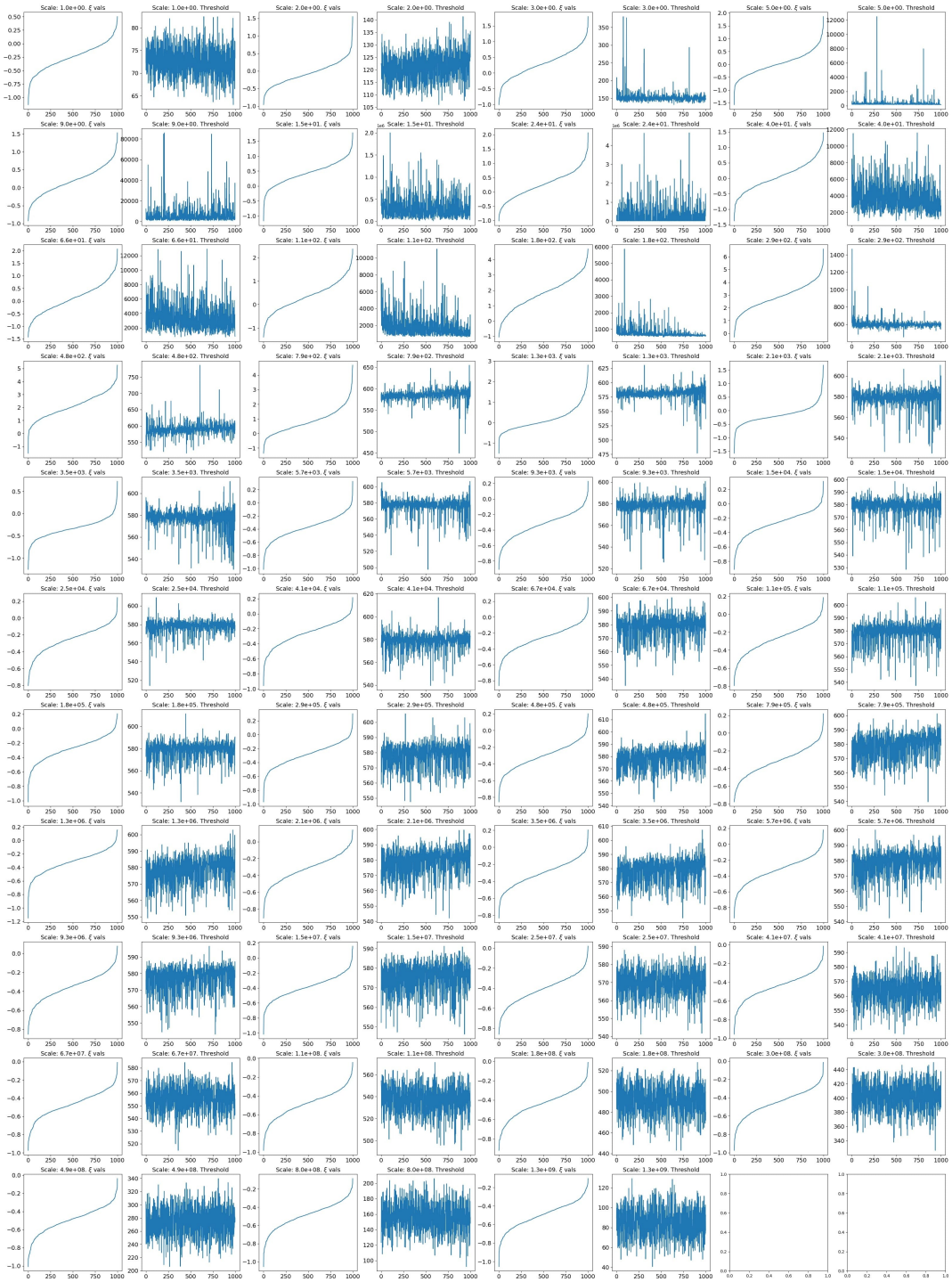

Fig. C.7.: For each length scale parameter of the Gaussian Process, we present the variability (sorted) of the estimated shape parameters across 1000 conditional distributions (defined by the choice of training sets). Jointly, we also present the 97th percentile of the conditional distributions corresponding to each estimated shape parameter.

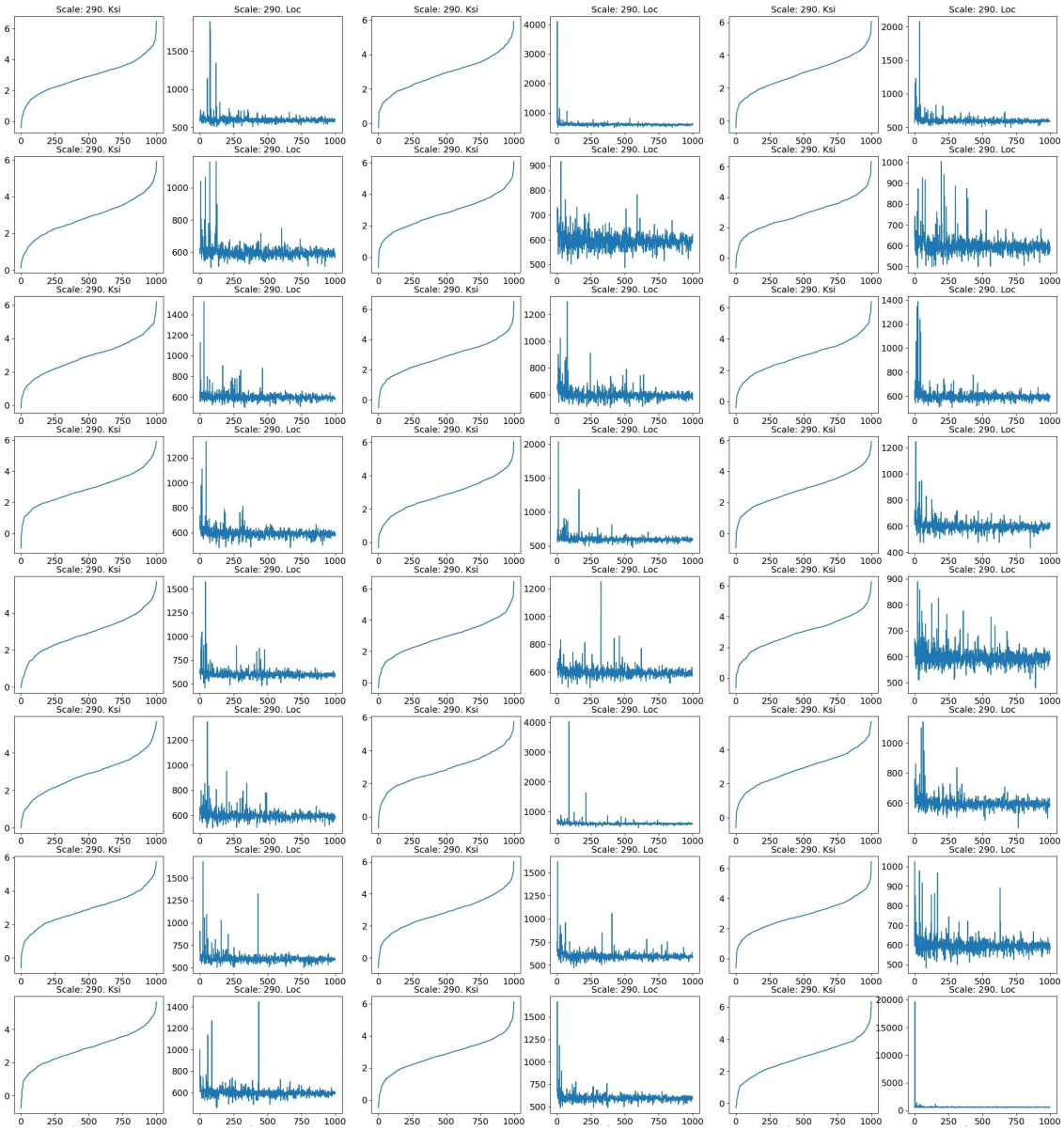

Fig. C.8.: We run 32 times the Gaussian Process experiment for length scale parameter value of 290. On each run, we calculate thresholds (sorted) of the 1000 conditional distributions determined by the 1000 choices of the training set, as well as their corresponding shape tail parameters. We see that higher thresholds correspond to lower shape parameters

C.9 Additional DEdH Tail Shape Estimator Experiments

Experimental validation of CTE using the DEdH estimator

A comprehensive set of experimental outcomes has been illustrated in Figure C.9.

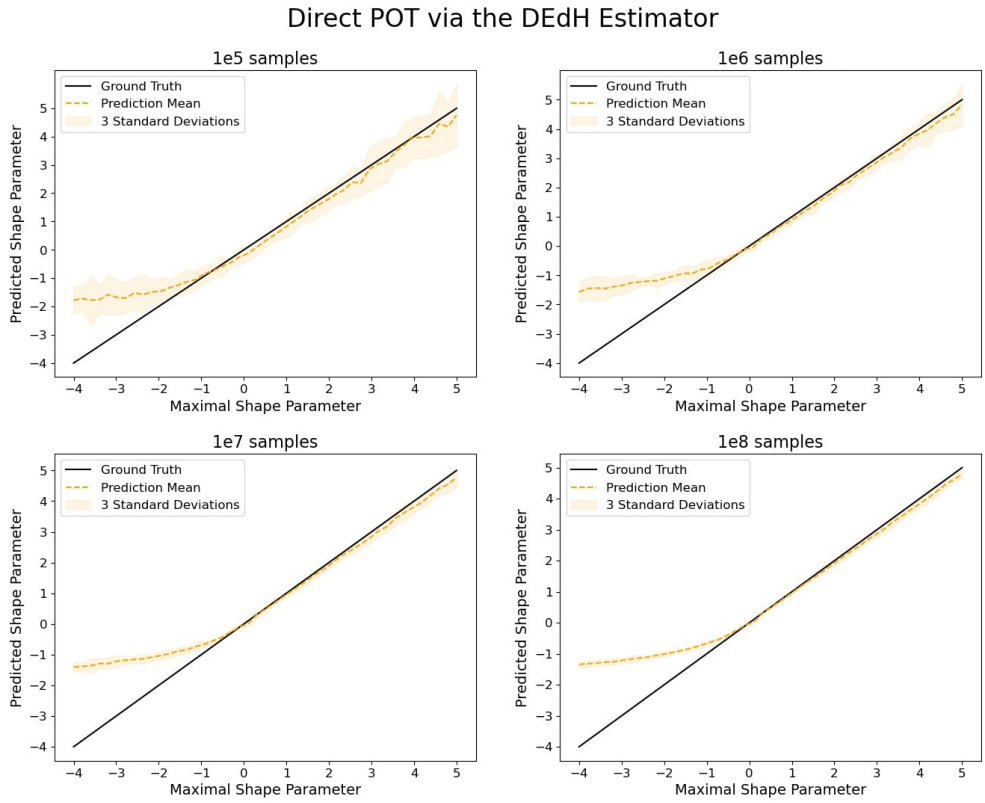

Fig. C.9.: In cases where the maximum tail shape parameter in the mixture of conditional distributions is positive, the estimated shape parameter of the marginal is also positive and equal to this maximal value. However, if this maximum value is negative, the estimated shape parameter is also negative. We utilize the DEdH estimator as our estimator of choice.

Just as in subsection 4.5.1, the parameter M, delineated in the subsection 4.5.1, is assigned values from the set $\{10^5, 10^6, 10^7, 10^8\}$. In the context of these experiments, p was set to 10 as a constant across all trials. The experiments were performed repetitively, encompassing a total of 10 runs to capture potential variability and better reflect the stochastic nature of the process.

Upon examining the obtained results, they again seem to align with our initial theoretical expectations. This symmetry in the estimations provides a degree of confidence in the validity of the conducted experiments and the consistency of the underlying theoretical framework.

Applying POT directly when the location of conditional distributions exhibits substantial variability

In the scope of our investigation, we executed experiments akin to those detailed in subsection 4.5.2, but in this case we utilized the DEdH estimator. Our observations reaffirm that the tail shape of the marginal is subject to incorrect estimations, which can

be attributed to the substantial variability in the location of the conditional distributions constituting the marginal.

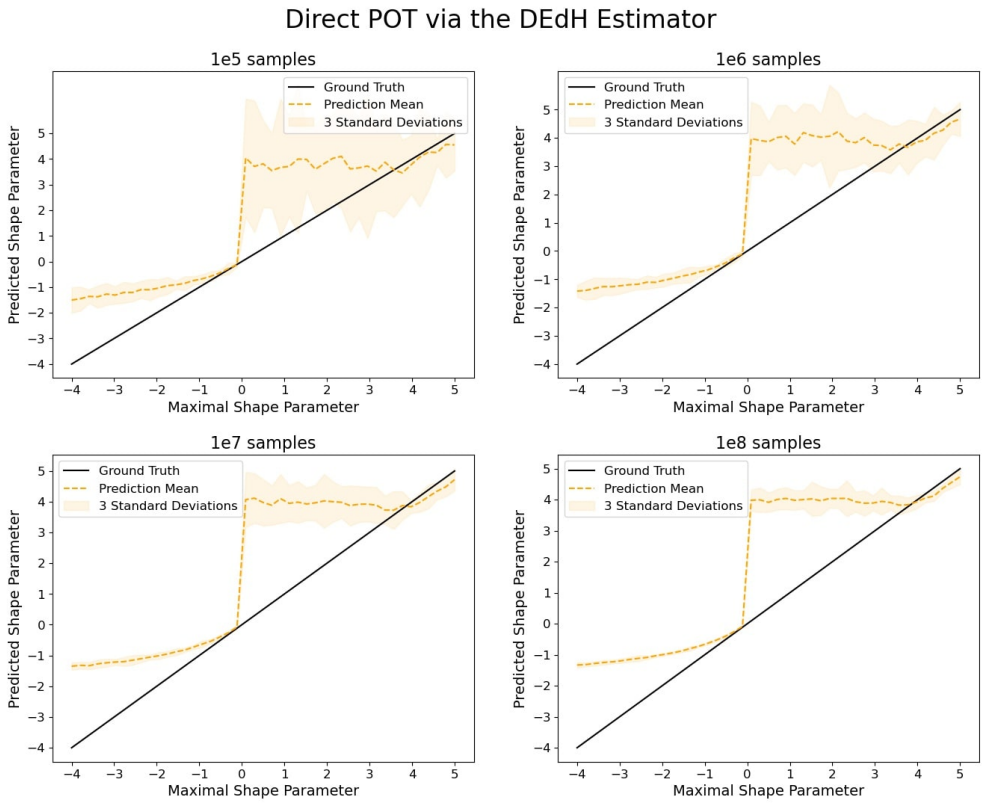

Fig. C.10.: Estimation of the shape parameter of the marginal by direct application of POT. We utilize the DEdH estimator as our estimator of choice.

Enhancing parameter estimation accuracy through the CTE approach

Analogous to the approach taken in section 4.5.2, we demonstrate here that the Cross Tail Estimation (CTE) effectively alleviates the issue associated with pronounced variation in the locations of conditional distributions that make up the marginal. It should be noted, however, that in this instance, we employ the DEdH estimator.

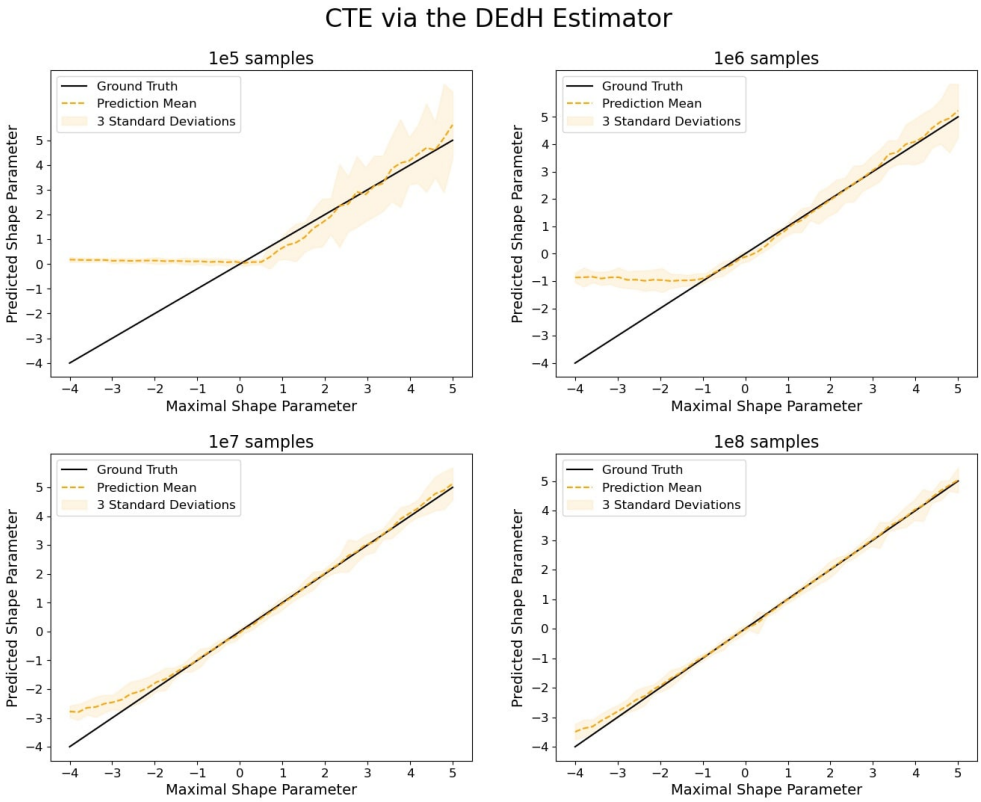

Fig. C.11.: Estimation the shape parameter of the marginal using CTE. We utilize the DEdH estimator as our estimator of choice.

List of Figures

List of Tables