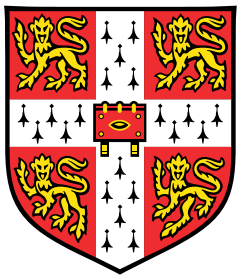
Bridging Deep Learning and Probabilistic Inference: Towards Data Efficiency, Identifiability, and Sampling Scalability



Wenlin Chen

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

St Edmund's College April 2025

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the preface and specified in the text. It is not substantially the same as any work that has already been submitted, or, is being concurrently submitted, for any degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee.

Wenlin Chen
April 2025

Abstract

While deep learning has achieved remarkable performance in modelling complex patterns in structured data, a key challenge is its reliance on large datasets. In contrast, probabilistic inference excels in data-scarce settings but suffers from computational inefficiencies for high dimensional data and struggles to model structured data where representation learning is crucial. This thesis focuses on the synergies between deep learning and probabilistic inference, bridging these gaps from two complementary perspectives, which results in novel machine learning methods with improved data efficiency, identifiability, and sampling scalability. In the first part of this thesis, we investigate how probabilistic inference can enhance deep learning. We first introduce a data-efficient meta-learning framework, which combines Gaussian processes and deep neural networks to improve representation learning on related low-data tasks. By formulating this problem in a novel bilevel optimisation framework and solving it with the implicit function theorem, this approach enhances the generalisation capabilities of deep neural networks for few-shot molecular property prediction and optimisation tasks. Next, we analyse the theoretical properties of neural network representations learned across multiple tasks within a probabilistic framework, establishing conditions under which neural networks can recover canonical feature representations that reflect the underlying ground-truth data generating process. Our framework not only ensures linear identifiability in the general multi-task regression setting, but also offers a simple probabilistic inference approach to recovering point-wise identifiable feature representations under certain assumptions of task structures, resulting in stronger theoretical guarantees and empirical performance of identifiability than previous methods on real-world molecular data. In the second part of this thesis, we explore the other direction in their reciprocal relationship: utilising deep learning to improve probabilistic inference. Inspired by diffusion-based modelling techniques, we propose a novel approach for training deep generative models to emulate sampling-based probabilistic inference for unnormalised probability distributions. This enables efficient sampling from multi-modal probability distributions such as Boltzmann distributions for many-body particle systems. Our approach outperforms previous neural samplers while achieving faster training and inference speed. Together, this thesis demonstrates how deep learning and probabilistic inference can be integrated in a mutually reinforcing manner to enhance each other.

Acknowledgements

First and foremost, I would like to thank my supervisor, Prof José Miguel Hernández-Lobato, for his unwavering support throughout my PhD. I am grateful for the freedom and encouragement that Miguel gave me to pursue my own ideas and explore different research directions. His critical feedback always challenged me to think deeper and helped me grow as a better researcher. I also want to thank my supervisor, Prof Bernhard Schölkopf, for his insightful comments on my work and for generously providing access to substantial GPU resources during my final year, which allowed me to freely explore interesting and ambitious ideas. I also appreciate the countless intriguing discussions that I had with my advisor, Dr Hong Ge, who taught me how to think outside the box when approaching research problems.

I am fortunate to have been part of the Computational and Biological Learning Lab at University of Cambridge and the Department of Empirical Inference at Max Planck Institute for Intelligent Systems, as they provide a truly wonderful and supportive environment for my PhD journey. I would like to thank the great collaborators I had the pleasure of working with during my PhD, from whom I learned a great deal: Austin Tripp, Mingtian Zhang, Jiajun He, Wen Wu, Wenlong Chen, Julien Horwood, Juyeon Heo, Zijing Ou, Yingzhen Li, RuiKang OuYang, and Severi Rissanen. I also want to extend my sincere thanks to Gregor Simm and Lixin Sun for offering an incredible internship experience at Microsoft Research.

I am grateful to everyone I met during my PhD who shared interesting conversations: Zongyu Guo, Rui Xia, Kaiqu Liang, Yanzhi Chen, Andy Lin, Isaac Reid, David Burt, Ross Clarke, Stratis Markou, Max Patacchiola, Vincent Dutordoir, James Allingham, Andrew Foong, Javier Antoran, Greg Flamich, Runa Eschenhagen, Bruno Mlodozeniec, Tor Fjelde, Adrian Goldwaser, Ross Viljoen, Yongchao Huang, Xianrui Zheng, Sukriti Singh, Laurence Midgley, Richard Turner, Zhenzhong Xiao, Annalena Kofler, Anson Lei, Weiyang Liu, Le Chen, Zeju Qiu, Siyuan Guo, Wendong Liang, Hsiao-Ru Pan, Gege Gao, Yamei Chen, Yaxi Hu, Florent Draye, Andreas Opedal, Andi Zhang, Zonghao Chen, Yu Xie, and many others. I also want to express my heartfelt thanks to Lexie for all the laughter and joy we have shared together.

Last but not least, I would like to express my deepest gratitude to my parents for their unconditional support throughout this journey.

Table of Contents

List of Figures

List of Tables

Nomenclature

Acronyms/Abbreviations

$\mathcal{W}$-2	Wasserstein-2
a.e.	Almost Everywhere
ADKF	Adaptive Deep Kernel Fitting
ADKL	Adaptive Deep Kernel Learning
AI	Artificial Intelligence
AIS	Annealed Importance Sampling
ARD	Automatic Relevance Determination
AUPRC	Area Under the Precision-Recall Curve
AUROC	Area Under the Receiver Operating Characteristic Curve
BNN	Bayesian Neural Network
BO	Bayesian Optimisation
CCA	Canonical Correlation Analysis
CMCD	Controlled Monte Carlo Diffusions
CNN	Convolutional Neural Network
CNP	Conditional Neural Process
const.	Constant

DDPM Denoising Diffusion Probabilistic Model

DDS Denoising Diffusion Sampler

DFT Density Functional Theory

diag Diagonal

DiGS Diffusive Gibbs Sampler

DiKL Diffusive Kullback-Leibler (Divergence)

DKL Deep Kernel Learning

DKT Deep Kernel Transfer

DSI Denoising Score Identity

DSM Denoising Score Matching

DW Double-Well

EC Enzyme Commission

EGNN Equivariant Graph Neural Network

EI Expected Improvement

EM Expectation-Maximisation

F-KL Forward Kullback-Leibler (Divergence)

FAB Flow Annealed Importance Sampling Bootstrap

GAN Generative Adversarial Network

GFlowNet Generative Flow Network

GNN Graph Neural Network

GP Gaussian Process

GPU Graphics Processing Unit

HMC Hamiltonian Monte Carlo

HOMO Highest Occupied Molecular Orbital Energy

i.i.d. Independently and Identically Distributed

ICA Independent Component Analysis

iCaRL Invariant Causal Representation Learning

iDEM Iterated Denoising Energy Matching

IFT Implicit Function Theorem

IS Importance Sampling

iVAE Identifiable Variational Autoencoder

KL Kullback-Leibler (Divergence)

kNN k-Nearest Neighbours

LG Langevin Dynamics

LJ Lennard-Jones

LSTM Long Short Term Memory

LUMO lowest Unoccupied Molecular Orbital Energy

MAC Multiply–Accumulate Operation

MALA Metropolis-Adjusted Langevin Algorithm

MAML Model-Agnostic Meta-Learning

MAT Molecule Attention Transformer

MCC Mean Correlation Coefficient

MCMC Markov Chain Monte Carlo

MH Metropolis-Hastings

MLE Maximum Likelihood Estimation

MLP Multi-Layer Perceptron

MoG Mixture of Gaussians

MSI Mixed Score Identity

MT Multi-Task

MTLCM Multi-Task Linear Causal Model

MTRN Multi-Task Regression Network

MW Many-Well

NETS Non-Equilibrium Transport Samplers

NLL Negative Log Likelihood

NLML Negative Log Marginal Likelihood

NLP Natural Language Processing

OOD Out-of-Domain/Distribution

PAR Property-Aware Relation Network

PCA Principal Component Analysis

PIS Path Integral Sampler

ProtoNet Prototypical Network

PT Parallel Tempering

R-DiKL Reverse Diffusive Kullback-Leibler (Divergence)

R-KL Reverse Kullback-Leibler (Divergence)

ReLU Rectified Linear Unit

RF Random Forest

RNN Recurrent Neural Network

SDE Stochastic Differential Equation

SiLU Sigmoid Linear Unit

SIR Sampling Importance Resampling

SKL Spread Kullback-Leibler (Divergence)

SM Score Matching

SNIS Self-Normalised Importance Sampling

ST Single-Task

TSI Target Score Identity

TVD Total Variation Distance

ULA Unadjusted Langevin Algorithm

VAE Variational Autoencoder

VI Variational Inference

VJP Vector-Jacobian Product

VP Variance-Preserving

VSD Variational Score Distillation

Chapter 1

Introduction

1.1 Motivation

Since the advent of AlexNet (Krizhevsky et al., 2012), deep learning has revolutionised many fields in computer science, such as computer vision (Betker et al., 2023; Dosovitskiy et al., 2021; He et al., 2016; Ho et al., 2020; Simonyan and Zisserman, 2015; Song et al., 2021b), natural language processing (Achiam et al., 2023; Devlin et al., 2019; Grattafiori et al., 2024; Guo et al., 2025; Mikolov et al., 2013; Vaswani et al., 2017) and speech processing (Baevski et al., 2020; Chen et al., 2022; Hinton et al., 2012; Hsu et al., 2021). More recently, this progress has extended beyond computer science research into the emerging interdisciplinary domain of *AI for science* (Wang et al., 2023a; Zhang et al., 2023b), which aims to harness the transformative power of deep learning to accelerate scientific discovery and tackle pressing societal challenges such as climate change, sustainable material design, and drug discovery.

One of the most compelling aspects of deep learning is its ability to directly produce high-quality solutions to scientific problems when provided with enormous amounts of training data, which far exceeds what traditional statistical data analysis approaches can offer. For example, deep-learning-based spatio-temporal models have demonstrated exceptional capability in simulating complex atmospheric dynamics with high resolution and precision thanks to the vast amounts of weather data, enabling accurate weather forecasting and extreme weather prediction across varying timescales (Allen et al., 2025; Bi et al., 2023; Bodnar et al., 2025; Lam et al., 2023; Pathak et al., 2022). Moreover, once trained, these models can be deployed on a personal laptop for real-time inference, which in many cases outperform traditional physics-based algorithms that require weeks of large-scale simulation on supercomputers.

However, deep learning's success heavily relies on access to large datasets for model training (Hestness et al., 2017). In low-data scenarios, deep neural networks are prone to overfitting,

often yielding highly confident yet inaccurate predictions (Gal, 2016; Goodfellow et al., 2016). In such cases, they may even underperform simple machine learning methods such as random forests or kernel-based approaches (Stanley et al., 2021). This presents a major barrier to applying deep learning to solve scientific problems where collecting high-quality training data requires costly experiments in laboratories or extensive simulations on supercomputers. For instance, in computational chemistry, deep-learning-based force field models have shown great promise in rapidly predicting the energies of molecules and the forces between atoms (Duval et al., 2023), but their accuracy heavily depends on the quantity and quality of training data. Generating high-fidelity data for training such models typically requires density functional theory (DFT) (Jones, 2015) simulation, which is computationally intensive and difficult to perform at scale.

In contrast, probabilistic inference offers a principled framework for developing robust machine learning models in the data-scarce regimes (Ghahramani, 2015; Murphy, 2012). However, it often struggles with high-dimensional, structured data where scalability and representation learning is crucial. Recognising the complementary strengths of deep learning and probabilistic inference, a growing body of research is attempting to integrate these two paradigms to enhance model reliability, accuracy, and efficiency. A prominent approach in this direction is Bayesian deep learning, which utilises Bayesian inference to estimate the parameters of deep neural networks, thereby enabling reliable uncertainty estimates in their predictions (Neal, 1996). These uncertainty estimates are particularly useful in scientific applications which involve decision-making under uncertainty, such as drug discovery where small datasets are ubiquitous and experimental validation is both costly and time-consuming. With access to the confidence estimates of model predictions, scientists can prioritise promising drug candidates while identifying molecules which warrant further wet-lab experimentation, ultimately reducing the overall costs and accelerating the process of drug discovery and development.

Importantly, the relationship between deep learning and probabilistic inference is *reciprocal*. While much prior work has focused on leveraging probabilistic inference to improve deep learning such as developing uncertainty quantification methods for deep neural networks, we highlight that deep learning also has the potential to improve the scalability and efficiency of probabilistic inference. For example, conventional molecular dynamics simulation approaches (Frenkel and Smit, 2023) typically rely on sampling-based probabilistic inference methods such as Markov chain Monte Carlo (MCMC), which can be prohibitively slow to generate equilibrium configurations for large protein complexes. In recent years, Boltzmann generators (Noé et al., 2019) have received significant interest in the sampling community, which employ deep learning to emulate molecular dynamics, offering an extremely powerful alternative to traditional sampling algorithms: once trained, these models can efficiently generate *independent* samples of protein configurations from Boltzmann distributions, significantly reducing

the time required for probabilistic inference from months to hours (Lewis et al., 2024). This demonstrates how deep learning can drastically accelerate probabilistic inference and provide scalable solutions to previously intractable scientific problems.

This thesis focuses on *synergistic* improvements in deep learning and probabilistic inference by leveraging their *bidirectional* relationship to enhance and advance methods from *both* fields. To evaluate our contributions in a meaningful and challenging setting, we focus on real-world molecular modelling tasks. This is a scientific application domain where neither pure deep learning models nor pure probabilistic inference approaches are fully effective on their own, since molecular data is high-dimensional, structured, and typically limited in quantity due to the high cost of laboratory experiments and computational simulations. Specifically, we consider the following two core applications in molecular science.

1. **Molecular property prediction.** We leverage probabilistic inference to enhance the *data efficiency* and *identifiability* of deep neural networks for improved representation learning. This results in improved generalisation performance and allows us to recover canonical feature representations for molecular property prediction tasks where many related small datasets are available.
2. **Molecular configuration sampling.** We employ generative deep learning to improve the *scalability* of sampling-based probabilistic inference. This enables efficient generation of equilibrium configurations from Boltzmann distributions of many-body particle systems without access to ground-truth samples.

1.2 Thesis Outline and Contributions

The outline of the remaining chapters in this thesis is as follows.

- Chapter 2 establishes the technical background for this thesis, introducing the key concepts and methods from probabilistic inference and deep learning. This chapter is concluded by reviewing two families of deep probabilistic models in the literature, demonstrating how methods from both fields can be combined effectively.
- Chapter 3 proposes a data-efficient meta-learning approach to extracting useful feature representations from related small datasets with deep kernel Gaussian processes, enabling probabilistic inference for task-specific layers in deep neural networks. This meta-learning problem is formulated within a novel bilevel optimisation framework, which can be solved efficiently by leveraging the implicit function theorem. The empirical performance of the proposed method is evaluated on few-shot molecular property

prediction and optimisation tasks, demonstrating its superior generalisation performance in low-data settings. This chapter is based on Chen et al. (2023).

- Chapter 4 further investigates the theoretical properties of the feature representations learned across multiple tasks with a probabilistic approach, particularly focusing on recovering canonical representations that agree with the underlying data generating process from observed data. We show that linear identifiability can be achieved with a regular multi-task regression neural network. Furthermore, point-wise identifiability may be achieved under certain assumptions of task structures. The new identifiability results presented in this chapter are validated on both synthetic tasks and real-world molecular property prediction tasks. This chapter is based on Chen et al. (2024a).
- Chapter 5 explores a complementary perspective to Chapter 3 and Chapter 4, aiming to enhance the efficiency of probabilistic inference with deep learning. Inspired by recent advances in generative deep learning, we propose a novel training paradigm for fitting deep generative models to unnormalised probability distributions with diffusion-inspired techniques. We demonstrate that the proposed algorithm enables us to train deep generative models that can efficiently produce accurate independent samples from both synthetic multi-modal distributions and Boltzmann distributions of many-body particle systems. This chapter is mainly based on He et al. (2025) and also contains materials from Chen et al. (2024b).
- Chapter 6 provides a comprehensive summary of our key findings and highlights the contributions made in this thesis. In addition, we identify several open questions and challenges, highlighting promising directions for future research.
- Appendices A-D provide supplementary materials for Chapters 2-5, including detailed experimental setups and model configurations, additional empirical evaluation results, and complete theoretical derivations.

1.3 List of Publications

This section provides a complete list of publications that I co-authored during my PhD upon submission of this thesis.

Peer-Reviewed Conference and Journal Publications

- (Chen et al., 2023) Wenlin Chen, Austin Tripp, José Miguel Hernández-Lobato. **Meta-Learning Adaptive Deep Kernel Gaussian Processes for Molecular Property Prediction**. *International Conference on Learning Representations (ICLR)*, 2023.

- (Chen et al., 2024a) Wenlin Chen*, Julien Horwood*, Juyeon Heo, José Miguel Hernández-Lobato. **Leveraging Task Structures for Improved Identifiability in Neural Network Representations.** *Transactions on Machine Learning Research (TMLR)*, 2024.
- (Chen et al., 2024b) Wenlin Chen*, Mingtian Zhang*, Brooks Paige, José Miguel Hernández-Lobato, David Barber. **Diffusive Gibbs Sampling**. *International Conference on Machine Learning (ICML)*, 2024.
- (Wu et al., 2024) Wen Wu*, Wenlin Chen*, Chao Zhang, Phil Woodland. **Modelling Variability in Human Annotator Simulation**. *Findings of the Association for Computational Linguistics (ACL)*, 2024.
- (Chen and Ge, 2024) Wenlin Chen, Hong Ge. **Neural Characteristic Activation Analysis and Geometric Parameterization for ReLU Networks**. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- (He et al., 2025) Jiajun He*, Wenlin Chen*, Mingtian Zhang*, David Barber, José Miguel Hernández-Lobato. **Training Neural Samplers with Reverse Diffusive KL Divergence**. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2025.
- (Rissanen et al., 2025) Severi Rissanen*, RuiKang OuYang*, Jiajun He, Wenlin Chen, Markus Heinonen, Arno Solin, José Miguel Hernández-Lobato. **Progressive Tempering Sampler with Diffusion**. *International Conference on Machine Learning (ICML)*, 2025.

Peer-Reviewed Workshop Publications

- (Tripp et al., 2022) Austin Tripp, Wenlin Chen, José Miguel Hernández-Lobato. **An Evaluation Framework for the Objective Functions of de novo Drug Design Benchmarks**. *Machine Learning for Drug Discovery (MLDD) Workshop at ICLR*, 2022.
- (Chen et al., 2025b) Wenlong Chen*, Wenlin Chen*, Lapo Rastrelli, Yingzhen Li. **Your Image is Secretly the Last Frame of a Pseudo Video**. *Deep Generative Model in Machine Learning: Theory, Principle and Efficacy (DeLTa) Workshop at ICLR*, 2025.
- (Zhang et al., 2025) Mingtian Zhang*, Wenlin Chen*, Jiajun He*, Zijing Ou, José Miguel Hernández-Lobato, Bernhard Schölkopf, David Barber. **Towards Training One-Step Diffusion Models Without Distillation**. *Deep Generative Model in Machine Learning: Theory, Principle and Efficacy (DeLTa) Workshop at ICLR*, 2025.

The asterisk superscript (*) indicates co-first authorship with equal contribution.

Chapter 2

Background

Deep learning and probabilistic inference are two rapidly evolving research fields in machine learning, each with a large body of literature. This chapter introduces the key concepts and methods in both fields that are relevant to the work presented in this thesis. We begin by introducing the basics of probabilistic inference (Section 2.1.1), using Gaussian processes (GPs) as an example where exact inference is tractable due to the analytical tractability of Gaussian distributions. Next, we review sampling-based inference methods (Section 2.1.2), which is a family of approximate inference methods commonly used when exact inference is intractable. We then proceed to introduce deep neural networks (Section 2.2.1) and explain how they can be used to extract feature representations from data (Section 2.2.2). Furthermore, we present two popular families of deep probabilistic models in the literature: Bayesian neural networks (Section 2.3.1) and deep generative models (Section 2.3.2), which integrate techniques from both fields, offering complementary perspectives on how deep learning and probabilistic inference can be combined effectively. Finally, we present some technical background on molecular machine learning, introducing common molecular representation methods for the two molecular modelling tasks considered in this thesis (Section 2.4).

2.1 Probabilistic Inference

Probabilistic inference offers a principled and powerful mathematical framework for expressing beliefs and quantifying uncertainty using probabilities when estimating unobserved variables. This framework allows us to incorporate prior knowledge into the inference process and facilitates continual updates as new information becomes available. This can be incredibly helpful in the scenarios where observations are limited or noisy, providing a solid foundation for robust and informed decision-making under uncertainty. This section reviews probabilistic

inference approaches relevant to this thesis. We refer the readers to Barber (2012); Bishop (2006); Murphy (2012) for a complete introduction on this topic.

2.1.1 Exact Inference

Rules of Probabilistic Inference

For a continuous random variable $x \in \mathcal{X}$, the probability density function $p(x)$ must satisfy the following two requirements:

- *Non-negative*:
$$p(x) \geq 0, \quad \forall x \in \mathcal{X}. \tag{2.1}$$
- *Integrating to one*:
$$\int p(x)\,\mathrm{d}x = 1. \tag{2.2}$$

For two random variables x and y with marginal probability density functions $p(x)$ and $p(y)$, conditional probability density functions $p(x|y)$ and $p(y|x)$, and joint probability density function $p(x, y)$, the two fundamental rules of probabilistic inference are as follows.

1. *Sum rule* reveals how to obtain marginal densities from the joint density:
$$\begin{aligned} p(x) &= \int p(x, y)\,\mathrm{d}y, \\ p(y) &= \int p(x, y)\,\mathrm{d}x. \end{aligned} \tag{2.3}$$

2. *Product rule* reveals how to obtain joint density from marginal and conditional densities:
$$p(x, y) = p(x|y)p(y) = p(y|x)p(x). \tag{2.4}$$

Note that x and y are *independent* if and only if their joint density factorizes into the product of the marginal densities:

$$p(x, y) = p(x)p(y) \quad \iff \quad x \perp\!\!\!\perp y. \tag{2.5}$$

Following Equation (2.3) and Equation (2.4), it is easy to derive the *Bayes' rule*:

$$p(y|x) = \frac{p(x, y)}{p(x)} = \frac{p(x|y)p(y)}{\int p(x|y)p(y)\,\mathrm{d}y} \propto p(x|y)p(y). \tag{2.6}$$

Bayesian inference utilises Bayes' rule to infer unobserved variables y from observed variables x, resulting in the *posterior* probability $p(y|x)$ which is proportional to the product of the *likelihood* $p(x|y)$ and the *prior* probability $p(y)$ of the unobserved variable y.

Below, we give an example of exact inference with Gaussian processes.

Gaussian Processes

A multivariate Gaussian distribution $\mathcal{N}(z|\mu, \Sigma)$ defines a probability distribution over a finite-dimensional variable $z \in \mathbb{R}^{d_z}$ ($d_z < \infty$), whose probability density function is given by

$$\mathcal{N}(z|\mu, \Sigma) \coloneqq (2\pi)^{-d_z/2} \det(\Sigma)^{-1/2} \exp\left(-\frac{1}{2}(z-\mu)^\top \Sigma^{-1}(z-\mu)\right), \tag{2.7}$$

where $\mu \in \mathbb{R}^{d_z}$ is a vector that defines the mean of the Gaussian distribution and $\Sigma \in \mathbb{R}^{d_z \times d_z}$ is a symmetric positive-definite matrix that defines the covariance of the Gaussian distribution.

Gaussian processes (GPs) (Rasmussen and Williams, 2006) can be viewed as infinite-dimensional generalisations of multivariate Gaussian distributions. A Gaussian process $\mathcal{GP}(m(\cdot), c(\cdot,\cdot))$ is fully specified by a mean function $m_\theta : \mathcal{X} \to \mathbb{R}$ and a symmetric positive-definite covariance function $c_\theta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

GPs offer a useful tool for specifying priors over functions. Consider a GP prior distribution over a function $f : \mathcal{X} \subseteq \mathbb{R}^{d_x} \to \mathbb{R}$:

$$p(f|\theta) = \mathcal{GP}(f|m_\theta(\cdot), c_\theta(\cdot,\cdot)), \tag{2.8}$$

where θ denotes the parameters of the mean function and covariance function. The covariance function $c_\theta(\cdot,\cdot)$ encodes the inductive bias (e.g., smoothness and periodicity) of the function specified by the GP. Following the convention of the GP literature (Neal, 1996), we will use zero mean function $m(\cdot) \equiv 0$ for GP priors throughout this thesis.

It is worth noting that evaluating the GP prior distribution over a function f at finite number of input locations $X = [x_1, \cdots, x_N]^\top \in \mathbb{R}^{N \times d_x}$ will result in a multivariate Gaussian distribution over the function values $\mathrm{f} \coloneqq f(X) \coloneqq [f(x_1), \cdots, f(x_N)]^\top \in \mathbb{R}^N$:

$$p(\mathrm{f}\,|X, \theta) = \mathcal{N}(\mathrm{f}\,|0, c_\theta(X, X)), \tag{2.9}$$

where the covariance matrix is obtained by evaluating the covariance function $c_\theta(\cdot,\cdot)$ at all pairs of input locations in X:

$$c_\theta(X,X) \coloneqq \begin{bmatrix} c_\theta(x_1,x_1) & c_\theta(x_1,x_2) & \cdots & c_\theta(x_1,x_N) \\ c_\theta(x_2,x_1) & c_\theta(x_2,x_2) & \cdots & c_\theta(x_2,x_N) \\ \vdots & \vdots & \ddots & \vdots \\ c_\theta(x_N,x_1) & c_\theta(x_N,x_2) & \cdots & c_\theta(x_N,x_N) \end{bmatrix} \in \mathbb{R}^{N\times N}. \tag{2.10}$$

For regression tasks, we assume that the corresponding observations $y \coloneqq [y_1,\cdots,y_N]^\top \in \mathbb{R}^N$ have an isotropic Gaussian likelihood with noise variance σ^2:

$$p(y|\,\mathrm{f},\theta) = \mathcal{N}(y|\,\mathrm{f},\sigma^2 I), \tag{2.11}$$

where the noise variance parameter σ^2 is included in θ for simplicity of notation.

One advantage of GP regression is that it is easy to perform principled model selection and obtain the closed-form probabilistic predictive distribution due to the analytical tractability of Gaussian distributions.

Specifically, we maximise the log marginal likelihood (or model evidence) of this GP regression model on the training dataset (X,y) with respect to all GP hyperparameters θ for model selection. The model evidence can be computed exactly as follows:

$$\begin{aligned} \log p(y|X,\theta) &= \log \int p(y|\,\mathrm{f},\theta)p(\mathrm{f}\,|X,\theta)\,\mathrm{d}\,\mathrm{f} && (2.12)\\ &= \log \mathcal{N}(y|0,c_\theta(X,X)+\sigma^2 I) && (2.13)\\ &= \underbrace{-\frac{1}{2}y^\top (c_\theta(X,X)+\sigma^2 I)^{-1}y}_{\text{data fit}} \underbrace{-\frac{1}{2}\log\det(c_\theta(X,X)+\sigma^2 I)}_{\text{model complexity}} + \text{const.} && (2.14) \end{aligned}$$

Interestingly, as shown in Equation (2.14), this objective regularises the GP regression model by automatically balancing between data fit and model complexity (Rasmussen and Williams, 2006).

After obtaining the optimal GP hyperparameters, we can make predictions for the test data X_*. By product rule, the posterior predictive distribution is defined as the ratio between the joint distribution and the marginal likelihood:

$$p(y_*|X_*,X,y,\theta) = \frac{p(y,y_*|X,x_*,\theta)}{p(y|X,\theta)} \propto p(y,y_*|X,x_*,\theta), \tag{2.15}$$

where the joint distribution is also a multivariate Gaussian distribution:

$$p(y, y_*|X, X_*, \theta) = \mathcal{N}\left(\begin{bmatrix} y \\ y_* \end{bmatrix} \middle| \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} c_\theta(X, X) + \sigma^2 I & c_\theta(X, X_*) \\ c_\theta(X_*, X) & c_\theta(X_*, X_*) + \sigma^2 I_* \end{bmatrix}\right). \tag{2.16}$$

Since the quotient of two Gaussian densities is a Gaussian density, we can obtain the mean and covariance of the posterior predictive distribution $p(y_*|X_*, X, y, \theta) \coloneqq \mathcal{N}(y_*|\mu_*, \Sigma_*)$ by matching the coefficients of y_* and $y_*^\top y_*$ in the joint density $p(y, y_*|X, X_*, \theta)$ and those in $p(y_*|X_*, X, y, \theta)$; see Murphy (2012) for a full derivation:

$$\mu_*(X_*, X, y) = c_\theta(X_*, X)\left(c_\theta(X, X) + \sigma^2 I\right)^{-1} y, \tag{2.17}$$

$$\Sigma_*(X_*, X) = c_\theta(X_*, X_*) + \sigma^2 I_* - \underbrace{c_\theta(X_*, X)\left(c_\theta(X, X) + \sigma^2 I\right)^{-1} c_\theta(X, X_*)}_{\text{uncertainty reduction}}. \tag{2.18}$$

Since computing the GP posterior predictive distribution involves inverting the covariance matrix $c_\theta(X, X)$ evaluated on the whole training dataset X, the computational cost of exact GP regression is $\mathcal{O}(N^3)$. This means that exact GP regression can be computationally intractable for large datasets. Fortunately, there exist approximate inference approaches for GPs based on inducing points, which significantly reduces the computational and memory costs; see, e.g., Titsias (2009) for more details.

Note that the uncertainty reduction in GP posterior predictive covariance in Equation (2.18) is independent of y and purely based on the similarity between the training inputs X and test inputs X_* measured by the covariance function, indicating that the GP regression prediction will be confident when the test input is close to some training inputs and uncertain otherwise.

Interestingly, the form of the GP posterior predictive distribution is reminiscent of Bayesian linear regression. In fact, GP regression is a function space representation of Bayesian linear regression of the form

$$f(x) = w^\top \phi(x) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2 I), \tag{2.19}$$

where the uncertainty of the function f is induced by a Gaussian prior over the regression weights $p_w(w) = \mathcal{N}(w|0, I)$ in the weight space, and the feature map $\phi : \mathcal{X} \to \mathbb{R}^{d_\phi}$ (typically $d_\phi > d_x$) is related to the GP covariance function:

$$c(x_i, x_j) \coloneqq \phi(x_i)^\top \phi(x_j); \tag{2.20}$$

see Rasmussen and Williams (2006) for a thorough discussion.

Regarding the choice of GP covariance functions in practice, the Matérn-ν kernel function is a family of covariance functions commonly used in GPs, which has the following form:

$$c_\theta^\nu(x_i, x_j) = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|x_i - x_j\|}{l} \right)^\nu K_\nu \left(\sqrt{2\nu} \frac{\|x_i - x_j\|}{l} \right), \tag{2.21}$$

where Γ is the gamma function, K_ν is the modified Bessel function of the second kind. The hyperparameters θ include a signal variance (i.e., amplitude) parameter σ_f^2 and a lengthscale parameter l. Instead of using a single lengthscale l for all input dimensions, one could also use *automatic relevance determination* (ARD), where a different lengthscale l_i is used for each input dimension $i = 1, \cdots, d_x$. The function specified by a GP with the Matérn-ν covariance function becomes smoother as we increase the value of ν. When $\nu = 1/2 + p$ for some $p \in \mathbb{N}$, the Matérn-ν kernel will have a simplified form.

- Matérn-$1/2$ kernel ($\nu = 1/2, p = 0$):

$$c_\theta^{1/2}(x_i, x_j) = \sigma_f^2 \exp\left(-\frac{\|x_i - x_j\|}{l} \right). \tag{2.22}$$

- Matérn-$3/2$ kernel ($\nu = 3/2, p = 1$):

$$c_\theta^{3/2}(x_i, x_j) = \sigma_f^2 \left(1 + \frac{\sqrt{3}\|x_i - x_j\|}{l} \right) \exp\left(-\frac{\sqrt{3}\|x_i - y_j\|}{l} \right). \tag{2.23}$$

- Matérn-$5/2$ kernel ($\nu = 5/2, p = 2$):

$$c_\theta^{5/2}(x_i, x_j) = \sigma_f^2 \left(1 + \frac{\sqrt{5}\|x_i - x_j\|}{l} + \frac{5\|x_i - x_j\|^2}{3l^2} \right) \exp\left(-\frac{\sqrt{5}\|x_i - x_j\|}{l} \right). \tag{2.24}$$

- As $\nu \to \infty$, the Matérn-ν kernel converges to the radial basis function (RBF) kernel:

$$\lim_{\nu \to \infty} c_\theta^\nu(x, y) = \sigma_f^2 \exp\left(-\frac{\|x_i - x_j\|^2}{2l^2} \right). \tag{2.25}$$

Tanimoto kernel (Ralaivola et al., 2005) is designed to handle binary vector inputs $x \in \{0, 1\}^{d_x}$, which is particularly useful to process binary molecular fingerprints (Tripp et al., 2023). It computes the Jaccard index (i.e., intersection over union) of the input binary vectors:

$$c_\theta^{\text{Tanimoto}}(x_i, x_j) = \frac{x_i^\top x_j}{1^\top x_i + 1^\top x_j - x_i^\top x_j}, \tag{2.26}$$

where $1 \in \mathbb{R}^{d_x}$ is a vector of ones. This kernel does not have any hyperparameters.

Finally, we note that new covariance functions can be constructed by combining existing covariance functions with the following operations.

- The *sum* of two covariance functions c_1, c_2 is a valid covariance function:

$$c(x_i, x_j) = c_1(x_i, x_j) + c_2(x_i, x_j). \tag{2.27}$$

- The *product* of two covariance functions c_1, c_2 is a valid covariance function:

$$c(x_i, x_j) = c_1(x_i, x_j)c_2(x_i, x_j). \tag{2.28}$$

- Multiplying a covariance function c_1 by a *positive scalar* $\alpha > 0$ results in a valid covariance function:

$$c(x_i, x_j) = \alpha c_1(x_i, x_j). \tag{2.29}$$

2.1.2 Sampling-Based Inference

When exact probabilistic inference is analytically and computationally intractable, approximate inference provides a feasible alternative. This section introduces sampling-based approximate inference methods relevant to this thesis. We refer the readers to Barber (2012) for a comprehensive review of other approximate inference approaches.

Monte Carlo Method

Consider an intractable target probability distribution

$$p_d(x) \propto \tilde{p}_d(x) \tag{2.30}$$

defined by an analytical but *unnormalised* probability density function $\tilde{p}_d(x)$ with an intractable normalising constant:

$$Z \coloneqq \int \tilde{p}_d(x)\,\mathrm{d}x. \tag{2.31}$$

This may be a Bayesian posterior distribution induced by a non-conjugate prior $\pi(x)$:

$$p_d(x) \coloneqq p(x|u) \propto p(u|x)\pi(x) \coloneqq \tilde{p}_d(x), \tag{2.32}$$

or a Boltzmann distribution defined by a non-trivial, lower-bounded energy function $E(x)$:

$$p_d(x) \propto \exp(-E(x)) \coloneqq \tilde{p}_d(x). \tag{2.33}$$

Probabilistic inference usually involves estimating the expectation of some function h of interest over the target distribution $p_d(x)$:

$$\mathbb{E}_{p_d(x)}[h(x)] \coloneqq \int h(x)p_d(x)\,\mathrm{d}x. \tag{2.34}$$

Although it is infeasible to obtain the exact probability density function $p_d(x)$ for the target distribution or analytically solve the integral in Equation (2.34), it is still possible to estimate this integral using the *Monte Carlo estimator* if we can draw samples from the target distribution:

$$\mathbb{E}_{p_d(x)}[h(x)] \approx \frac{1}{L}\sum_{l=1}^{L} h(x^{(l)}) \coloneqq \hat{h}, \tag{2.35}$$

where $x^{(1:L)} \sim \hat{p}_d(x^{(1:L)})$ are approximate samples of $p_d(x)$ from the sampling distribution $\hat{p}_d(x^{(1:L)})$. If all marginal sampling distributions match the target distribution, i.e., $\hat{p}_d(x^{(l)}) = p_d(x)$ for all $l = 1, \cdots, L$, then the Monte Carlo estimator is unbiased:

$$\mathbb{E}_{\hat{p}_d(x^{(1:L)})}[\hat{h}] = \frac{1}{L}\sum_{l=1}^{L} \mathbb{E}_{\hat{p}_d(x^{(l)})}[h(x^{(l)})] = \mathbb{E}_{p_d(x)}[h(x)], \tag{2.36}$$

and the variance of the Monte Carlo estimator is given by

$$\mathrm{Var}_{\hat{p}_d(x^{(1:L)})}[\hat{h}] = \frac{1}{L}\mathrm{Var}_{p_d(x)}[h(x)] + \frac{1}{L^2}\sum_{l\neq l'} \mathrm{Cov}_{\hat{p}_d(x^{(l)},x^{(l')})}[h(x^{(l)}), h(x^{(l')})]. \tag{2.37}$$

Furthermore, if the samples are *independently and identically distributed* (i.i.d.):

$$\hat{p}_d(x^{(1:L)}) = \prod_{l=1}^{L} \hat{p}_d(x^{(l)}) = \prod_{l=1}^{L} p_d(x^{(l)}), \tag{2.38}$$

then the second term on the right hand side of Equation (2.37) vanishes, and hence the variance of the Monte Carlo estimator reduces to

$$\mathrm{Var}_{\hat{p}_d(x^{(1:L)})}[\hat{h}] = \frac{1}{L}\mathrm{Var}_{p_d(x)}[h(x)]. \tag{2.39}$$

Therefore, with i.i.d. samples from $p_d(x)$, the error of the Monte Carlo estimator shrinks at rate $\mathcal{O}(1/\sqrt{L})$, which is *independent* of the dimensionality d_x of the random variable x.

Below, we review some common sampling algorithms.

Importance Sampling

Importance sampling (IS) uses samples from a simple proposal distribution $q(x)$ to estimate the expectation $\mathbb{E}_{p_d(x)}[h(x)]$ as defined in Equation (2.34). The proposal distribution must have a tractable probability density function $q(x)$ and allow for efficient exact sampling (e.g., Gaussian distributions). The IS estimator works as follows:

$$\int h(x)p_d(x)\,\mathrm{d}x = \int h(x)\frac{p_d(x)}{q(x)}q(x)\,\mathrm{d}x \tag{2.40}$$

$$:= \int h(x)w_{\text{IS}}(x)q(x)\,\mathrm{d}x \tag{2.41}$$

$$\approx \frac{1}{L}\sum_{l=1}^{L} h(x^{(l)})w_{\text{IS}}(x^{(l)}), \tag{2.42}$$

where $x^{(1:L)} \sim q(x)$ are i.i.d. samples from the proposal distribution $q(x)$, and the IS weights are defined as the density ratio between the target distribution and proposal distribution:

$$w_{\text{IS}}(x) := \frac{p_d(x)}{q(x)}. \tag{2.43}$$

For unnormalised target distributions $p_d(x) = \tilde{p}_d(x)/Z$, we can additionally apply IS to estimate the normalising constant Z, which results in a self-normalised importance sampling (SNIS) estimator:

$$\int h(x)\frac{\tilde{p}_d(x)}{Z}\,\mathrm{d}x = \frac{\int h(x)\tilde{p}_d(x)\,\mathrm{d}x}{\int \tilde{p}_d(x)\,\mathrm{d}x} \tag{2.44}$$

$$:= \frac{\int h(x)\tilde{w}_{\text{IS}}(x)q(x)\,\mathrm{d}x}{\int \tilde{w}_{\text{IS}}(x)q(x)\,\mathrm{d}x} \tag{2.45}$$

$$\approx \frac{\sum_{l=1}^{L} h(x^{(l)})\tilde{w}_{\text{IS}}(x^{(l)})}{\sum_{l'=1}^{L} \tilde{w}_{\text{IS}}(x^{(l')})} \tag{2.46}$$

$$:= \sum_{l=1}^{L} h(x^{(l)})w_{\text{SNIS}}(x^{(l)}) \tag{2.47}$$

where $x^{(1:L)} \sim q(x)$ are i.i.d. samples from the proposal distribution $q(x)$, the IS weights are defined with the unnormalised target density function $\tilde{p}_d(x)$ in this case:

$$\tilde{w}_{\text{IS}}(x) := \frac{\tilde{p}_d(x)}{q(x)}, \tag{2.48}$$

and the SNIS weights are obtained by normalising the IS weights:

$$w_{\text{SNIS}}(x^{(l)}) := \frac{\tilde{w}_{\text{IS}}(x^{(l)})}{\sum_{l'=1}^{L} \tilde{w}_{\text{IS}}(x^{(l')})}. \tag{2.49}$$

Intuitively, each IS weight $w(x^{(l)})$ re-weights the function value $h(x^{(l)})$ evaluated at the corresponding IS sample $x^{(l)} \sim q(x)$ according to its contribution to the expectation. For IS to be efficient, the proposal distribution $q(x)$ should have high density in the regions where $p_d(x)$ has high density and $h(x)$ has large magnitude, since such proposal distributions minimise the variance of the IS estimator; see Owen (2013) for more details.

It is worth noting that IS is inefficient for high dimensional random variables, since the density of a high dimensional distribution usually concentrates in a very small region of the support. As a result, even a slight mismatch between $q(x)$ and $p_d(x)$ can lead to most IS samples having negligible contributions to the estimate of the expectation. To make the best use of available IS samples, one can alternatively perform sampling importance resampling (SIR), which resamples the IS samples $x^{(1:L)}$ with replacement according to the empirical distribution specified by the SNIS weights, reallocating samples with low importance weights to those with high importance weights. The resampled set of samples follows the target distribution $p_d(x)$, which can be used in the standard Monte Carlo estimator as defined in Equation (2.35).

Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods are a family of sampling algorithms which draw a sequence of *dependent* samples from the target distribution $p_d(x)$ by simulating a first-order Markov chain that has $p_d(x)$ as its unique equilibrium distribution (i.e., it will eventually converge to the target distribution $p_d(x)$ regardless of the initial state). Note that there are many Markov chains whose equilibrium distributions are $p_d(x)$, and therefore one can design different MCMC samplers by constructing different Markov transition kernels $T(x'|x)$.

It can be shown that a Markov chain has a unique equilibrium distribution $p_d(x)$ if

1. it is *irreducible* (i.e., it can go from any state x to any other state x' in finite steps);
2. it is *aperiodic* (i.e., it does not periodically revisit any state x);
3. it leaves the target distribution $p_d(x)$ *invariant*, i.e.,

$$p_d(x') = \int T(x'|x) p_d(x) \, \mathrm{d}x. \tag{2.50}$$

Note that a Markov transition kernel $T(x'|x)$ is a conditional probability density function which must be positive and integrate to one for any given x. One way to construct a Markov chain that leaves $p_d(x)$ invariant is to find a Markov transition kernel that satisfies the *detailed balance* condition (and thus the corresponding Markov chain is reversible):

$$T(x'|x) p_d(x) = T(x|x') p_d(x'), \quad \forall x, x', \tag{2.51}$$

which implies that $p_d(x)$ is the invariant distribution:

$$\int T(x'|x)p_d(x)\,\mathrm{d}x = \int T(x|x')p_d(x')\,\mathrm{d}x = p_d(x')\int T(x|x')\,\mathrm{d}x = p_d(x'). \tag{2.52}$$

Metropolis-Hastings Algorithm

The Metropolis-Hastings (MH) algorithm (Hastings, 1970; Metropolis et al., 1953) provides a simple way to construct a Markov transition kernel $T(x'|x)$ that satisfies the detailed balance condition from any proposal distribution $q(x'|x)$ with an additional acceptance step:

$$T(x'|x) = A(x'|x)q(x'|x), \tag{2.53}$$

where the acceptance rate $A(x'|x)$ is defined as

$$A(x'|x) \coloneqq \min\left(1, \frac{q(x|x')p_d(x')}{q(x'|x)p_d(x)}\right) = \min\left(1, \frac{q(x|x')\tilde{p}_d(x')}{q(x'|x)\tilde{p}_d(x)}\right). \tag{2.54}$$

It is easy to verify that $T(x'|x)$ always satisfies the detailed balance condition. Note that the MH algorithm only requires evaluating the unnormalised target density $\tilde{p}_d(x)$, since the normalising constant is cancelled out from the numerator and demonstrator in Equation (2.54). In practice, the MH algorithm works as follows at each step:

1. Propose a new state $x' \sim q(x'|x)$ given the current state x.
2. Calculate the acceptance rate $\alpha_{\text{MH}} \coloneqq A(x'|x)$ according to Equation (2.54).
3. Accept the proposal x' as the next state with probability α_{MH}. If the proposal is rejected, retain the current state x as the next state.

It is importance to note that the choice of the proposal distribution $q(x'|x)$ determines the performance of the MCMC sampler, and the optimal choice of $q(x'|x)$ depends on the properties of the landscape of the target distribution $p_d(x)$. Simple proposals such as Gaussian random walk $q(x'|x) = \mathcal{N}(x'|x, \sigma_r^2 I)$ is unlikely to perform well for complicated high-dimensional target distributions due to its "blindness": it typically requires a tiny variance σ_r^2 to maintain a reasonable acceptance rate (Bishop, 2006), resulting in highly correlated samples. Below, we introduce two classes of advanced MCMC algorithms with adaptive proposals that leverage the local landscape information of the target density function.

Langevin Dynamics

Metropolis-adjusted Langevin Algorithm (MALA) (Roberts and Stramer, 2002; Roberts and Tweedie, 1996) defines its proposal distribution based on a discrete-time Langevin stochastic

differential equation (SDE):

$$q(x'|x) = \mathcal{N}(x'|x + \eta\nabla_x \log \tilde{p}_d(x), \sqrt{2\eta}I), \tag{2.55}$$

where $\eta > 0$ is a hyperparameter that controls the step size, and the discretisation bias can be corrected by the MH algorithm. As the step size $\eta \to 0$, the discretisation errors vanish and thus the MH correction step may be omitted, which recovers the unadjusted Langevin algorithm (ULA) (Grenander and Miller, 1994; Roberts and Tweedie, 1996).

Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) (Duane et al., 1987; Hoffman et al., 2014; Neal et al., 2011) augments the original random variable x with an auxiliary momentum variable v, which defines a joint probability distribution over the phase space:

$$p(x, v) \coloneqq p_d(x)p(v) \propto \tilde{p}_d(x)e^{-K(v)}, \tag{2.56}$$

where $p(v) \coloneqq \mathcal{N}(v|0, M)$ corresponds to the kinetic energy $K(v) \coloneqq \frac{1}{2}v^\top M^{-1}v$ with a hyperparameter M that represents the mass matrix. The Hamiltonian of this system is given by $H(x, v) \coloneqq -\log \tilde{p}_d(x) + K(v)$. HMC generates (x, v) samples by simulating the following Hamiltonian equations:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{\partial H}{\partial v} = M^{-1}v, \tag{2.57}$$

$$\frac{\mathrm{d}v}{\mathrm{d}t} = -\frac{\partial H}{\partial x} = \nabla_x \log \tilde{p}_d(x). \tag{2.58}$$

Accurate numerical simulation of the Hamiltonian equations can be done by the leapfrog algorithm (Neal et al., 2011). For a given current state x, a single step of the leapfrog algorithm with step size η proceeds as follows:

$$v \sim \mathcal{N}(0, M), \tag{2.59}$$

$$v_{\eta/2} = v + \frac{\eta}{2}\nabla_x \log \tilde{p}_d(x), \tag{2.60}$$

$$x' = x + \eta M^{-1}v_{\eta/2}, \tag{2.61}$$

$$v' = v_{\eta/2} + \frac{\eta}{2}\nabla_{x'} \log \tilde{p}_d(x'). \tag{2.62}$$

In practice, we typically perform several steps of the leapfrog algorithm to generate a new sample. In theory, the acceptance rate of the HMC proposal is always one since the Hamiltonian is conserved in a Hamiltonian system (i.e., $H(x, v) \equiv H(x', v')$) and the Hamiltonian dynamics is always reversible. Nevertheless, the MH algorithm is usually employed to correct

the potential discretisation bias in practice:

$$\alpha_{\text{HMC}} \equiv \min\left\{1, \frac{\exp(-H(x', v'))}{\exp(-H(x, v))}\right\}. \tag{2.63}$$

Annealed Importance Sampling

The initial state of an MCMC sampler is often a random sample from some simple initial distribution $q(x)$. Although the Markov chain will eventually converge to the target distribution $p_d(x)$ regardless of the initial state, it can take a very long time for the MCMC sampler to mix well in practice when $q(x)$ is far away from $p_d(x)$. Also, recall that in IS , the expectations of functions over $p_d(x)$ are estimated with samples from a simple proposal distribution $q(x)$ which may not have many overlaps with $p_d(x)$, resulting in inefficient sampling.

Annealed importance sampling (AIS) (Neal, 2001) introduces a sequence of intermediate distributions $\{\pi_{(k)}(x)\}_{k=1}^{K}$ that interpolate between the proposal distribution $q(x)$ and the target distribution $p_d(x)$ to facilitate a smoother transition:

$$\pi_{(k)}(x) \propto q(x)^{1-\beta_k} p_d(x)^{\beta_k} \propto q(x)^{1-\beta_k} \tilde{p}_d(x)^{\beta_k}, \tag{2.64}$$

where $0 = \beta_0 < \beta_1 < \cdots < \beta_{K-1} < \beta_K = 1$, $\pi_{(0)}(x) \coloneqq q(x)$ is the proposal distribution, and $\pi_{(k)}(x) \coloneqq p_d(x)$ is the target distribution. The AIS algorithm runs iteratively as follows:

- Draw the initial sample $x_{(0)} \sim \pi_{(0)}(x) = q(x)$ from the proposal distribution.
- For $k = 1, 2, \cdots, K-1$, draw $x_{(k)} \sim \pi_{(k)}(x)$ by running an MCMC sampler (e.g., HMC) whose initial state is the previous sample $x_{(k-1)}$ and which leaves the current intermediate distribution $\pi_{(k)}$ invariant.

In the end, AIS produces a sample $x \coloneqq x_{(K-1)}$ which is close to the target distribution $\pi_{(K)}(x) = p_d(x)$. We can then calculate the IS weight in the joint space over $x_{(0:K-1)}$, which defines the AIS weight:

$$\tilde{w}_{\text{AIS}}(x_{(0:K-1)}) \propto \prod_{k=1}^{K} \frac{\pi_{(k)}(x_{(k-1)})}{\pi_{(k-1)}(x_{(k-1)})}. \tag{2.65}$$

Finally, we can obtain the (self-normalised) AIS estimator by replacing the IS samples $x^{(1:L)} \sim q(x)$ and IS weights $\tilde{w}_{\text{IS}}$ in the SNIS estimator as defined in Equation (2.47) with the

AIS samples $x_{(0:K-1)}^{(1:L)} \sim \text{AIS}$[1] and AIS weights $\tilde{w}_{\text{AIS}}$:

$$\int h(x)p_d(x)\,\mathrm{d}x \approx \frac{\sum_{l=1}^{L} h(x^{(l)})\tilde{w}_{\text{AIS}}(x_{(0:K-1)}^{(l)})}{\sum_{l'=1}^{L} \tilde{w}_{\text{AIS}}(x_{(0:K-1)}^{(l')})}. \tag{2.66}$$

Alternatively, one can also perform SIR with AIS samples and weights to obtain a resampled set of samples that follow the target distribution $p_d(x)$.

2.2 Deep Learning

This section reviews neural network architectures and representation learning approaches relevant to this thesis. We refer the readers to Bishop and Bishop (2023); Goodfellow et al. (2016) for a comprehensive introduction to deep learning.

2.2.1 Deep Neural Networks

Deep neural networks are a large class of highly flexible non-linear models capable of extracting intrinsic patterns in data. This enables them to model complex mappings from inputs to outputs. Different neural network architectures incorporate distinct inductive biases tailored to specific types of input data. For instance, convolutional neural networks (CNNs) (LeCun et al., 1998) excel in modelling spatial data (e.g., image), recurrent neural networks (RNNs) (Cho et al., 2014; Hochreiter and Schmidhuber, 1997) and transformers (Vaswani et al., 2017) are well-suited for sequential data (e.g., text and audio), and graph neural networks (GNNs) (Scarselli et al., 2008) are designed to process relational data (e.g., graph). Below, we introduce two types of neural network architectures relevant to this thesis.

Multi-Layer Perceptrons

Multi-layer perceptrons (MLPs) are the most generic neural network architectures with fully connected neurons, which are suitable for general function approximation. Mathematically, an MLP layer applies an affine transformation to the input $x \in \mathbb{R}^{d_{\text{in}}}$ followed by an element-wise non-linear activation function g, which produces an output $x' \in \mathbb{R}^{d_{\text{out}}}$:

$$x' = g(W^\top x + b), \tag{2.67}$$

where the weight matrix $W \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ and the bias vector $b \in \mathbb{R}^{d_{\text{out}}}$ are learnable parameters, and d_{in} and d_{out} are the dimensionalities of the input and output vectors, respectively. In

[1] We use superscripts with parentheses to denote sample index and subscripts with parentheses to denote the index of each intermediate step in AIS.

practice, a single MLP layer may not be sufficiently flexible. To better capture the intricate patterns in complex data, we typically need to construct *deep* neural networks by stacking multiple MLP layers on top of one another depending on the complexity of the function to be approximated.

In modern neural network architectures, Rectified Linear Unit (ReLU) (Glorot et al., 2011) and its variants are the most widely used activation functions due to their simplicity and effectiveness. The ReLU activation function is a linear function for positive inputs and outputs zero for negative inputs:

$$\text{ReLU}(s) \coloneqq \max(0, s). \tag{2.68}$$

Sigmoid Linear Unit (SiLU) (Hendrycks and Gimpel, 2016) is a variant of ReLU which avoids the hard cut-off point at $s = 0$ by using a smoother transition defined by the sigmoid function:

$$\text{SiLU}(s) \coloneqq s \odot \text{Sigmoid}(s) = \frac{s}{1 + \exp(-s)}, \tag{2.69}$$

where $\odot$ denotes the element-wise multiplication operator.

Graph Neural Networks

Graph neural networks (GNNs) are well-suited for extracting feature representations from graph data such as molecules (Bruna et al., 2014; Corso et al., 2020; Duvenaud et al., 2015; Gilmer et al., 2017; Gori et al., 2005; Jin et al., 2017; Kearnes et al., 2016; Kipf and Welling, 2017; Li et al., 2016; Scarselli et al., 2008; Schütt et al., 2017; Xu et al., 2019).

In a GNN, the node features $\mathcal{V} = \{v_m\}_{m=1}^{M}$ and edge features $\mathcal{E} = \{e_{ij}\}_{i,j=1}^{M}$ of a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ are iteratively updated in each layer. Predictions can then be made by passing the global feature representation u of the graph $\mathcal{G}$ to a readout network. Below, we describe a general formulation of GNNs following Battaglia et al. (2018).

A GNN layer updates each node feature v_m by *message passing*:

$$v'_m = \texttt{UPDATE}\left(v_m, \texttt{AGGREGATE}\left(\{(v_m, v_j, e_{mj}) | j \in N(m)\}\right)\right), \tag{2.70}$$

where $N(m)$ is a set that contains indices of nodes that are neighbours of node m, `AGGREGATE` is a set function parameterised by a neural network that is permutation invariant to the elements in a set, and `UPDATE` is an MLP network which produces updated node features v'_m from the current node features v_m and the aggregated messages from its neighbour nodes. A graph-level global feature representation u is then obtained by applying a set function `READOUT` to the set of all node features from the final GNN layer:

$$u = \texttt{READOUT}\left(\{v_m^{(\text{final})}\}_{m=1}^{M}\right). \tag{2.71}$$

Deep set (Zaheer et al., 2017) provides a simple way to construct set functions using neural networks: it applies a single neural network to every element in the set, aggregates the resulting feature representations using a permutation invariant operation such as averaging, and processes the aggregated representation with an MLP network.

Moreover, for tasks which involve modelling the 3D positions of atoms in the Cartesian coordinate, a specialised class of GNNs called geometric graph neural networks is designed to ensure that their outputs remain *invariant* or *equivariant* to the translations and rotations of the input molecules (Batatia et al., 2022; Satorras et al., 2021), inherently respecting the symmetries of physical systems. This eliminates the need for GNNs to learn to model every possible invariant or equivariant configuration, significantly improving data-efficiency and model generalisation. We refer the readers to Duval et al. (2023) for a comprehensive survey of geometric deep learning for graph data.

2.2.2 Supervised Representation Learning

Deep neural networks are powerful tools for extracting informative low-dimensional feature representations from high-dimensional structured data, which are crucial for generalisation, interpretability, and transferability (Bengio et al., 2013). In supervised learning, we typically optimise the input-output mapping performance of deep neural networks. Once trained, the learned representations that capture the features of the input data at different levels of abstractions can be derived from the intermediate layers of these models.

Multi-Task Representation Learning

Deep neural networks often require a large amount of training data to learn useful representations (Goodfellow et al., 2016). However, small datasets are ubiquitous in many scientific problems, making it challenging for deep neural networks to learn reliable representations from any single dataset alone. Alternatively, if we have access to many *related* datasets which share some common mechanisms, we may be able to extract feature representations that are useful for simultaneously solving all these tasks (Caruana, 1997). A typical multi-task representation learning approach employs a shared deep neural network backbone to produce a common representation, followed by a task-specific output layer (or "head") that works on top of the common representation to solve each task. This particular setup encourages the backbone to encode information that is useful for all tasks, which promotes knowledge transfer across all datasets and reduces overfitting on any individual dataset during training.

Identifiability

Since modern deep neural networks are over-parameterised by design, the feature representations learned by these models are generally unidentifiable (Roeder et al., 2021). This means that even with sufficient training data, and even if our model family contains the ground-truth model and the training loss converges to the global optimum value, there is no guarantee that the learned representations will reflect the underlying ground-truth data generating process. It is desirable to recover canonical feature representations, since they may offer valuable insights for understanding the underlying mechanisms in many scientific problems. The lack of identifiability limits the interpretability and out-of-distribution generalisation capabilities of the learned representations (Lu et al., 2022). Therefore, establishing conditions under which deep neural networks can recover identifiable representations is an active research field.

2.3 Probabilistic Deep Learning

This section reviews two widely used classes of deep probabilistic models: Bayesian neural networks and deep generative models, which are representative examples from the literature that aim to combine the strengths of probabilistic inference and deep neural networks from two complementary perspectives.

2.3.1 Bayesian Neural Networks

For deep neural networks trained on small datasets, it is crucial to quantify the uncertainty in their predictions, especially for downstream tasks that involve decision-making (Antoran, 2024; Gal, 2016). Bayesian neural networks (BNNs) address this by treating neural network parameters as random variables and using Bayesian inference to update their probability distributions (Barber and Bishop, 1998; Hernández-Lobato and Adams, 2015; Neal, 1996). Specifically, for a given training set $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$, the posterior density of the parameters θ of a neural network $f_\theta(x)$ can be obtained by Bayes' rule:

$$p(\theta|\mathcal{D}) \propto \prod_{n=1}^{N} p(y_n|x_n, \theta) p_\theta(\theta), \tag{2.72}$$

where $p_\theta(\theta)$ is the prior density of the parameters, and $p(y_n|x_n, \theta)$ is the likelihood parameterised by the neural network (e.g., a conditional Gaussian distribution $\mathcal{N}(y|f_\theta(x), \sigma^2)$ for regression tasks). Since exact inference for Equation (2.72) is generally intractable due to the non-linearity of the neural network $f_\theta(x)$, approximate inference techniques such as MCMC (Neal, 1996), amortised variational inference (Blundell et al., 2015) and Laplace's approximation (Daxberger et al., 2021) are commonly employed. Finally, the posterior predictive

distribution of a neural network can be obtained by marginalising out the parameters over their posterior density from the likelihood:

$$p(y_*|x_*, \mathcal{D}) = \int p(y_*|x_*, \theta)p(\theta|\mathcal{D})\,\mathrm{d}\theta \tag{2.73}$$

$$\approx \frac{1}{L}\sum_{l=1}^{L} p(y_*|x_*, \theta^{(l)}), \quad \theta^{(1:L)} \sim p(\theta|\mathcal{D}). \tag{2.74}$$

However, balancing computational tractability while maintaining high-quality of uncertainty estimates remains a major challenge in practice (Antoran, 2024; Fortuin, 2022; Tomczak et al., 2021). Instead of adopting a fully Bayesian approach, an alternative in Bayesian deep learning is to perform Bayesian inference only for the parameters in the last layer of a neural network (Harrison et al., 2024; Lázaro-Gredilla and Figueiras-Vidal, 2010; Ober and Rasmussen, 2019; Watson et al., 2021). This is often referred to as a Bayesian last layer, which strikes a practical trade-off between reliable uncertainty estimation and computational scalability. This approach is also closely related to deep kernel learning (Hinton and Salakhutdinov, 2007; Wilson et al., 2016b), where a GP is placed on top of a deep neural network to capture the uncertainty in the output layer of the neural network.

2.3.2 Deep Generative Models

Deep generative models leverage deep neural networks to improve the quality of generative modelling for data distributions of interest. Below, we introduce two types of deep generative models relevant to this thesis: deep latent variable models and diffusion models.

Deep Latent Variable Models

Deep latent variable models are a family of deep generative models which define the marginal density of data $x \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$ through a latent variable:

$$p_\theta(x) = \int p_\theta(x|z)p_z(z)\,\mathrm{d}z, \tag{2.75}$$

where the latent variable $z \in \mathcal{Z} \subseteq \mathbb{R}^{d_z}$ ($d_z \leq d_x$) follows a simple distribution $p_z(z)$ apriori such as the standard Gaussian distribution, and the likelihood $p_\theta(x|z)$ is a conditional distribution parameterised by a neural network with learnable parameters θ. For a target distribution $p_d(x)$ with i.i.d. ground-truth samples $\mathcal{D} = \{x_n\}_{n=1}^N$, deep latent variables are often trained by maximising the log marginal density $\log p_\theta(x)$ averaged over the training set $\mathcal{D}$ with respect to θ. Below, we introduce some common deep latent variable models.

Variational autoencoders (VAEs) (Kingma and Welling, 2013) pre-define the likelihood, e.g., using a Gaussian distribution $p_\theta(x|z) = \mathcal{N}(x|\mu_\theta(z), \sigma^2 I)$ with its mean parameterised by a neural network $\mu_\theta(z)$ and a fixed variance σ^2, which results in an intractable model density $p_\theta(x)$. Kingma and Welling (2013); Wainwright et al. (2008) derive a tractable variational lower bound of the log marginal density $\log p_\theta(x)$ with a learnable variational distribution $q_\phi(z|x)$ for training VAEs:

$$\log p_\theta(x) = \log \int p_\theta(x|z) p_z(z) \, \mathrm{d}z \tag{2.76}$$

$$= \log \mathbb{E}_{q_\phi(z|x)} \left[\frac{p_\theta(x|z) p_z(z)}{q_\phi(z|x)} \right] \tag{2.77}$$

$$\geq \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p_\theta(x|z) p_z(z)}{q_\phi(z|x)} \right] \coloneqq \mathcal{F}(\theta, \phi), \tag{2.78}$$

where the last line follows by Jensen's inequality and the equality holds when the approximate posterior becomes exact:

$$q_\phi(z|x) = p_\theta(z|x) \coloneqq \frac{p_\theta(x|z) p_z(z)}{p_\theta(x)}. \tag{2.79}$$

Note that the variational lower bound $\mathcal{F}(\theta, \phi)$ can be expressed as a linear combination of a reconstruction error term for the observed variable x and a KL regularisation term for the latent variable z, which is the origin of the name *variational autoencoders*:

$$\mathcal{F}(\theta, \phi) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathrm{KL}(q_\phi(z|x)||p_z(z)) \tag{2.80}$$

$$= -\frac{1}{2\sigma^2} \underbrace{\mathbb{E}_{q_\phi(z|x)}[\|x - \mu_\theta(z)\|^2]}_{\text{reconstruction error of } x} - \underbrace{\mathrm{KL}(q_\phi(z|x)||p_z(z))}_{\text{KL regularisation of } z} + \text{const.}, \tag{2.81}$$

The variational distribution is usually chosen to be a mean-field Gaussian distribution:

$$q_\phi(z|x) \coloneqq \mathcal{N}(z|m_\phi(x), \mathrm{diag}(s_\phi(x)^2)) \tag{2.82}$$

whose mean $m_\phi(x)$ and variance $s_\phi(x)^2$ are parameterised by a neural network with learnable parameters ϕ. In practice, both sets of parameters θ and ϕ are jointly trained by maximising the lower bound $\mathcal{F}(\theta, \phi)$ of the log marginal density. The expectation over $q_\phi(z|x)$ is estimated by the Monte Carlo method with the reparametrisation trick (Kingma and Welling, 2013):

$$z = m_\phi(x) + s_\phi(x) \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I), \tag{2.83}$$

which results in a differentiable sample from the variational distribution $q_\phi(z|x)$, since the standard Gaussian noise ε is independent of the parameters θ and ϕ. While this variational

approach circumvents the intractability of $\log p_\theta(x)$, it introduces its own challenges and limitations, such as the limited flexibility of the variational family, the potential looseness of the variational bound, and the variational over-pruning issue (Turner and Sahani, 2011).

Normalising flows (Dinh et al., 2017; Kingma and Dhariwal, 2018; Papamakarios et al., 2021; Rezende et al., 2020) employ a deterministic likelihood with an invertible neural network f_θ, which results in a tractable model density that can be directly optimised:

$$\log p_\theta(x) = \log p_z(f_\theta^{-1}(x)) + \log \left| \det \left(\frac{\partial f_\theta^{-1}(x)}{\partial x} \right) \right|. \tag{2.84}$$

More generally, deep implicit models, such as generative adversarial networks (GANs) (Goodfellow et al., 2014; Nowozin et al., 2016), also use a Dirac delta likelihood function $\delta(\cdot)$ but allow the generator f_θ to be a standard non-invertible neural network (Goodfellow et al., 2014; Huszár, 2017):

$$p_\theta(x) = \int \delta(x - f_\theta(z)) p_z(z) \, \mathrm{d}z, \tag{2.85}$$

which is more flexible than VAEs and normalising flows, as it avoids pre-defining a constrained distribution family for the likelihood $p_\theta(x|z)$ and does not require the generator $f_\theta(z)$ to be invertible. However, the model density $p_\theta(x)$ of a deep implicit model may not have a valid density function. For example, when $d_z < d_x$, the resulting model density $p_\theta(x)$ may not be absolutely continuous (Arjovsky et al., 2017; Zhang et al., 2020). In this case, we refer to $p_\theta(x)$ as *generalised* model density. For this reason, we may not be able to train implicit models with common likelihood-based parameter estimation approaches. For instance, GANs employ adversarial training instead with an additional discriminator network.

Diffusion Models

Diffusion models have achieved photo-realistic visual generation quality (Ho et al., 2020; Karras et al., 2022; Sohl-Dickstein et al., 2015; Song et al., 2021a). Unlike deep latent variable models which directly map a latent variable z to observed data x, diffusion models employ a sequence of intermediate variables $x_1, x_2, \cdots, x_T \in \mathbb{R}^{d_x}$ to capture the features of data x[2] at different fidelity and facilitate a gradual generation path from noise to data:

$$p_\theta(x) = \int p_z(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \, \mathrm{d}x_{1:T}. \tag{2.86}$$

Diffusion models employ a forward diffusion process to create a sequence of increasingly noisy data $x_1, \cdots, x_T$ by progressively adding small Gaussian noise until the data x is corrupted to white noise x_T. On the distributional level, the forward diffusion process gradually transforms

[2]For simplicity of notation, we define $x_0 \coloneqq x$.

the data distribution $p_d(x)$ to a known noise distribution $p_z(x_T)$ (e.g., standard Gaussian distribution) with a sequence of pre-defined Gaussian convolution kernels:

$$k_t(x_t|x) \coloneqq \mathcal{N}(x_t|\alpha_t x, \sigma_t^2 I), \quad t = 1, \dots, T. \tag{2.87}$$

The resulting marginal distribution of noisy data at each time t is given by

$$p_{d,t}(x_t) = \int k_t(x_t|x) p_d(x) \, \mathrm{d}x. \tag{2.88}$$

Given the forward diffusion process, the generation model $p_\theta(x_{t-1}|x_t)$ essentially defines a *denoising* process, which generates a data $x \sim p_d(x)$ by gradually denoising a white noise $x_T \sim p_z(x_T)$. In practice, we typically model the denoising process by

$$p_\theta(x_{t-1}|x_t) \coloneqq \int k_{t-1}(x_{t-1}|x, x_t) p_\theta(x|x_t) \, \mathrm{d}x, \tag{2.89}$$

where the denoising posterior is approximated by a conditional Gaussian distribution

$$p_\theta(x|x_t) \coloneqq \mathcal{N}(x|\mu_\theta(x_t, t), \Sigma_t), \tag{2.90}$$

with mean $\mu_\theta(x_t, t)$ parameterised by a time-conditioned neural network with learnable parameters θ and covariance $\Sigma_t = \tilde{\sigma}_t^2 I$ fixed to some pre-defined constant value[3]. Song et al. (2021a) show that the smoothing distribution of the following functional form corresponds to the forward diffusion process defined in Equation (2.87):

$$k_{t-1}(x_{t-1}|x, x_t) = \mathcal{N}\left(x_{t-1} \middle| \alpha_{t-1} x_0 + \frac{x_t - \alpha_t x_0}{\sqrt{1 - \alpha_t^2}} \sqrt{1 - \alpha_{t-1}^2 - s_t^2}, s_t^2 I\right), \tag{2.91}$$

where the value of s_t^2 determines the properties of the diffusion process (e.g., Markov property and stochasticity).

There are various parameterisations for diffusion models. Karras et al. (2022) directly work with the denoising mean parametrisation of diffusion models by training a time-conditioned denoising network $\mu_\theta(x_t, t)$ to predict the denoising mean of the data x from its noisy version x_t for all t with the EDM loss:

$$\mathcal{L}_{\text{EDM}}(\theta) = \mathbb{E}_{\mathcal{U}(t) p_d(x) k_t(x_t|x)}[\lambda(t) \|\mu_\theta(x_t, t) - x\|^2], \tag{2.92}$$

where $\mathcal{U}(t) \coloneqq \mathcal{U}(t|\{1, \cdots, T\})$ denotes the uniform distribution over $t \in \{1, \cdots, T\}$, $\lambda(t)$ is a positive scalar weighting function, and the expectation is estimated by the Monte Carlo

[3] It is possible to estimate the optimal covariance of the denoising posterior distribution, which can lead to better denoising performance (Bao et al., 2022; Ou et al., 2025).

method. Once the denoising network $\mu_\theta(x_t, t)$ is trained, we can simulate the denoising process in Equation (2.89) using the Monte Carlo method with samples $x \sim p_\theta(x|x_t) = \mathcal{N}(x|\mu_\theta(x_t, t), \Sigma_t)$ from the denoising posterior distribution.

Interestingly, Tweedie's formula (Efron, 2011; Robbins, 1992) reveals an important connection between the denoising posterior mean $\mu_\theta(x_t, t)$ and the noisy model score $s_\theta(x_t, t) := \nabla_{x_t} \log p_{\theta,t}(x_t)$:

$$\mu_\theta(x_t, t) = \frac{x_t + \sigma_t^2 s_\theta(x_t, t)}{\alpha_t}, \tag{2.93}$$

which allows us to learn the denoising posterior mean $\mu_\theta(x_t, t)$ from data using denoising score matching (DSM) (Vincent, 2011).

Proposition 2.3.1 (Denoising Score Identity). *For any convolution kernel $k_t(x_t|x)$, we have*

$$\nabla_{x_t} \log p_{d,t}(x_t) = \int \nabla_{x_t} \log k_t(x_t|x) p_d(x|x_t) \, \mathrm{d}x, \tag{2.94}$$

where $p_d(x|x_t) \propto k_t(x_t|x)p_d(x)$ is the denoising posterior distribution. For $k_t(x_t|x) = \mathcal{N}(x_t|\alpha_t x, \sigma_t^2 I)$ as defined in Equation (2.87), this recovers Tweedie's formula.

The proof of Proposition 2.3.1 can be found in Appendix A.1. Note that Proposition 2.3.1 also holds for the noisy model score $\nabla_{x_t} \log p_{\theta,t}(x_t)$.

Song and Ermon (2019); Song et al. (2021b) work with the score parametrisation of diffusion models by training a time-conditioned score network $s_\theta(x_t, t)$ to predict the noisy data score $\nabla_{x_t} \log p_{d,t}(x_t)$ for all t with the DSM loss:

$$\mathcal{L}_{\text{DSM}}(\theta) = \mathbb{E}_{\mathcal{U}(t)p_d(x)k_t(x_t|x)}[\lambda(t)\|s_\theta(x_t, t) - \nabla_{x_t} \log k_t(x_t|x)\|^2]. \tag{2.95}$$

The derivation of the DSM loss function is based on the score matching (SM) loss (Hyvärinen, 2005) and DSI, which can be found in Appendix A.2. Once the score model $s_\theta(x_t, t)$ is trained, we can plug it into Tweedie's formula as stated in Equation (2.93) to obtain the denoising posterior mean $\mu_\theta(x_t, t)$.

Alternatively, Ho et al. (2020); Song et al. (2021a) work with the noise parametrisation of diffusion models by training a time-conditioned noise prediction network $\varepsilon_\theta(x_t, t)$ to predict the noise ε used to corrupt the data x to its noisy version x_t for all t with the DDPM loss:

$$\mathcal{L}_{\text{DDPM}}(\theta) = \mathbb{E}_{\mathcal{U}(t)p_d(x)\mathcal{N}(\varepsilon|0,I)}[\lambda(t)\|\varepsilon_\theta(\alpha_t x + \sigma_t \varepsilon, t) - \varepsilon\|^2]. \tag{2.96}$$

Once the noise prediction model $\varepsilon_\theta(x_t, t)$ is trained, the denoising posterior mean $\mu_\theta(x_t, t)$ can be obtained by

$$\mu_\theta(x_t, t) = \frac{x_t - \sigma_t \varepsilon_\theta(x_t, t)}{\alpha_t}. \tag{2.97}$$

2.4 Molecular Representations for Machine Learning

As molecules are structured data, they must be encoded into machine-readable representations suitable for processing by machine learning models. This section introduces molecular representation methods commonly used in machine learning for the two molecular modelling tasks addressed in this thesis.

2.4.1 Molecular Property Prediction

Molecular property prediction is the task that aims to predict the properties (e.g., toxicity and permeability) of molecules from their structures. Deep learning approach has become popular for this task in recent years due to its fast inference speed and competitive accuracy (Gilmer et al., 2017), which has the potential to accelerate the drug discovery process (Stanley et al., 2021). The following three molecular representation methods are frequently employed in machine learning.

1. *Simplified Molecular Input Line Entry System* (SMILES) (Weininger, 1988, 1990; Weininger et al., 1989) is a line notation for representing molecular graphs using strings following a set of fixed rules. These strings are easily readable by humans and can be processed by standard natural language processing (NLP) models such as LSTM (Hochreiter and Schmidhuber, 1997). One undesirable property of SMILES strings is that they may transform short-range dependencies between atoms in a molecular graph into long-range dependencies in the corresponding SMILES string in some cases.

2. *Molecular fingerprints* (Rogers and Hahn, 2010) encode molecular graphs into vector representations where each element corresponds to a molecular substructure. The algorithm takes a molecular graph, a radius parameter R, and a vector length parameter L as input. It first applies a hash function to atom features in the graph to assign integers to atoms. Then, for each radius $r = \{1, \cdots, R\}$, it concatenates atom integers with the integers of their neighbouring atoms, and assigns new integers to atoms by applying the hash function to the concatenations. Finally, the molecular fingerprint representation is obtained by creating an L-dimensional vector of zeros and setting every entry that is mapped to a generated integer for an atom to one.

3. *Graph representation* of molecules is natural with minimal loss of information, since molecular graphs are invariant to permutation of atoms and can encode distances between atoms. A molecule with M atoms can be represented by an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where node features $\mathcal{V} = \{v_m\}_{m=1}^{M}$ are associated with atoms, and edge features $\mathcal{E} = \{e_{ij}\}_{i,j=1}^{M}$ are associated with bonds. Molecular graphs can be processed by GNNs.

2.4.2 Molecular Configuration Sampling

Molecular configuration sampling is the task that aims to generate all possible 3D structures of a given molecule from its Boltzmann distribution, which is essential for simulating or emulating molecular dynamics and estimating key macroscopic physical quantities (e.g., binding affinity). Deep learning approach has attracted significant attention for this task in recent years due to its potential to accelerate and amortise this process (e.g., emulating the dynamics of large proteins) (Noé et al., 2019). The following two coordinate systems are often used to represent the 3D structures of molecules in machine learning.

1. *Internal coordinates* describe 3D molecular structures using bond lengths, bond angles and dihedral angles. This compact representation has less redundancy and thus is typically computationally more efficient and easier to manipulate. However, internal coordinates vary across different molecules, making it challenging to transfer learned models between them.

2. *Cartesian coordinates* represent 3D molecular structures using ordered-triples (x, y, z) in three-dimensional space. This representation is intuitive for humans and consistent across all molecules. However, it poses challenges for machine learning models, as it requires accounting for physical constraints such as invariance or equivariance to certain transformations.

Chapter 3

Meta-Learning Gaussian Processes for Data-Efficient Representation Learning

This chapter is based on Chen et al. (2023):

- Wenlin Chen, Austin Tripp, José Miguel Hernández-Lobato. **Meta-Learning Adaptive Deep Kernel Gaussian Processes for Molecular Property Prediction**. *International Conference on Learning Representations (ICLR)*, 2023.

All co-authors co-developed the method and co-wrote the manuscript. In addition, I wrote all code and performed all experiments for this work.

As introduced and discussed in the previous chapter, deep neural networks are powerful tools for learning useful feature representations from data for downstream tasks (Bengio et al., 2013). However, the "data-hungry" nature limits their performance in low-data regimes (Gal, 2016; Goodfellow et al., 2016). On the other hand, Gaussian processes (GPs) (Rasmussen and Williams, 2006) are well-calibrated probabilistic models with generally reliable uncertainty on small datasets. Deep kernel GPs (Patacchiola et al., 2020; Wilson et al., 2016b), which combines neural network representations and GP models, have the potential to improve the robustness of neural networks on small datasets. However, previous methods for training these models often lead to unsatisfactory results due to overfitting or underfitting issues. This chapter presents Adaptive Deep Kernel Fitting with Implicit Function Theorem (ADKF-IFT), a novel framework for training deep kernel GPs by interpolating between meta-learning and conventional deep kernel learning. This approach employs a bilevel optimisation objective, with the resulting nested optimisation problem solved by the implicit function theorem (IFT). We show that the proposed ADKF-IFT framework contains previous methods for training deep kernel GPs as special cases. We find that ADKF-IFT is especially well-suited for low-

data tasks such as drug discovery, and demonstrate that it significantly outperforms previous state-of-the-art methods on a variety of real-world few-shot molecular property prediction tasks and out-of-domain molecular property prediction and optimisation tasks.

3.1 Motivation and Overview

Many real-world applications require machine learning algorithms to make robust predictions with well-calibrated uncertainty given very limited training data. One important example is drug discovery, where practitioners not only want models to accurately predict biochemical and physicochemical properties of molecules, but also want to use models to guide the search for novel molecules with desirable properties, leveraging techniques such as Bayesian optimisation (BO) which heavily rely on accurate uncertainty estimates (Frazier, 2018). Despite the meteoric rise of neural networks over the past decade, their notoriously overconfident and unreliable uncertainty estimates (Szegedy et al., 2014) make them generally ineffective surrogate models for BO. Instead, most contemporary BO implementations use Gaussian processes (GPs) (Rasmussen and Williams, 2006) as surrogate models due to their analytically tractable and generally reliable uncertainty estimates, even on small datasets.

Traditionally, GPs are fit on hand-engineered features (e.g., molecular fingerprints), which can limit their predictive performance on complex, structured, high-dimensional data such as molecules where designing informative features is challenging. Naturally, a number of works have proposed to improve performance by instead fitting GPs on features learned by a deep neural network: a family of models generally called Deep Kernel GPs. However, there is no clear consensus about how to train these models: maximising the GP marginal likelihood (Hinton and Salakhutdinov, 2007; Wilson et al., 2016b) has been shown to overfit on small datasets (Ober et al., 2021), while meta-learning (Patacchiola et al., 2020) and fully-Bayesian approaches (Ober et al., 2021) avoid this at the cost of making strong, often unrealistic assumptions. This suggests that there is demand for new, better techniques for training deep kernel GPs.

This chapter presents a novel, general framework called *Adaptive Deep Kernel Fitting with Implicit Function Theorem* (ADKF-IFT) for training deep kernel GPs, which is especially well-suited to small datasets. ADKF-IFT essentially trains a subset of the model parameters with a meta-learning loss, and separately adapts the remaining parameters on each task using maximum marginal likelihood estimation. In contrast to previous methods which use a single loss for all parameters, ADKF-IFT is able to utilise the implicit regularisation of meta-learning to prevent overfitting while avoiding the strong assumptions of a pure meta-learning approach which may lead to underfitting. The key contributions of this chapter are summarised as follows.

1. As our main technical contribution, we present the general ADKF-IFT framework and its natural formulation as a bilevel optimisation problem (Section 3.3.1), then explain how the implicit function theorem (IFT) can be used to efficiently solve it with gradient-based methods in a few-shot learning setting (Section 3.3.2).
2. We show how ADKF-IFT can be viewed as a *generalisation* and *unification* of previous approaches based purely on single-task learning (Wilson et al., 2016b) or purely on meta-learning (Patacchiola et al., 2020) for training deep kernel GPs (Section 3.3.3).
3. We propose a specific practical instantiation of ADKF-IFT wherein all feature extractor parameters are meta-learned, which has a clear interpretation and obviates the need for any Hessian approximations. We argue why this particular instantiation is well-suited to retain the best properties of previously proposed methods (Section 3.3.4).
4. Motivated by the general demand for better GP models in chemistry, we perform an extensive empirical evaluation of ADKF-IFT on several chemical tasks, finding that it significantly improves upon previous state-of-the-art methods (Section 3.5).

3.2 Preliminaries

3.2.1 Few-Shot Learning

Few-shot learning (Vinyals et al., 2016) refers to learning on many related tasks when each task has few labelled examples (Lake et al., 2011; Miller et al., 2000). It requires learning algorithms to be able to adapt to unseen tasks given few labelled examples. Few-shot learning algorithms often benefit from knowledge transfer among related low-data tasks, whereas single-task learning algorithms can easily overfit or underfit in this setting. In the few-show learning setup, one is given a set of training tasks $\mathcal{D} = \{\mathcal{T}_t\}_{t=1}^{T}$ (a *meta-dataset*) and some unseen test tasks $\mathcal{D}_* = \{\mathcal{T}_*\}$. Each task $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^{N_\mathcal{T}}$ is a set of points in the domain $\mathcal{X}$ (e.g., space of molecules) with corresponding labels (continuous, categorical, etc.), and is partitioned into a *support set* $\mathcal{S}_\mathcal{T} \subseteq \mathcal{T}$ for meta-training and a *query set* $\mathcal{Q}_\mathcal{T} = \mathcal{T} \setminus \mathcal{S}_\mathcal{T}$ for meta-testing. Typically, the total number of training tasks $|\,\mathcal{D}\,|$ is large, while the size of each support set $|\,\mathcal{S}_\mathcal{T}\,|$ is small. Models for few-shot learning are typically trained to accurately predict unseen examples in the query set $\mathcal{Q}_\mathcal{T}$ given labelled examples in the support set $\mathcal{S}_\mathcal{T}$ for all tasks $\mathcal{T} \in \mathcal{D}$ during a *meta-training* phase, then evaluated by their prediction error on $\mathcal{Q}_{\mathcal{T}_*}$ given $\mathcal{S}_{\mathcal{T}_*}$ for unseen test tasks $\mathcal{T}_* \in \mathcal{D}_*$ during a *meta-testing* phase.

3.2.2 Deep Kernel Gaussian Processes

Deep Kernels

Deep Kernel Gaussian Processes are GPs whose covariance function is constructed by first using a neural network *feature extractor* f_ϕ with parameters $\phi \in \Phi$ to create feature representations $h = f_\phi(x), h' = f_\phi(x')$ of the input points x, x', then feeding these feature representations into a standard *base kernel* $c_\theta(h, h')$ (e.g., an RBF kernel) (Bradshaw et al., 2017; Calandra et al., 2016; Hinton and Salakhutdinov, 2007; Wilson et al., 2016a,b). The complete covariance function is therefore $k_\psi(x, x') \coloneqq c_\theta(f_\phi(x), f_\phi(x'))$ with learnable parameters $\psi \coloneqq (\theta, \phi)$. Deep kernel GPs can be viewed as performing Bayesian inference for the last linear layer of a neural network in the function space (Harrison et al., 2024; Lázaro-Gredilla and Figueiras-Vidal, 2010; Ober and Rasmussen, 2019; Watson et al., 2021).

Deep Kernel Learning

Deep Kernel Learning (DKL) (Wilson et al., 2016b) is a single-task method for fitting a deep kernel to a dataset. DKL jointly fits both the feature extractor parameters ϕ and base kernel parameters θ by maximising the GP marginal likelihood on a single dataset (i.e., DKL essentially fits a neural network with a GP "head" to a dataset).

It is well known that neural networks will easily overfit to small datasets (Sarle, 1995). This overfitting also happens in DKL, despite the fact that it fits the neural network parameters using a type-II maximum likelihood approach: although early DKL papers suggested that the "model complexity" term (as measured by the log determinant of the kernel matrix) in the GP marginal likelihood objective as stated in Equation (2.14) would prevent this overfitting from happening (Wilson et al., 2016b), recent follow-up work showed that this is not the case (Ober et al., 2021) as a deep-kernel GP can simultaneously overfit to the training data and appear to have a low "model complexity".

Deep Kernel Transfer

Deep Kernel Transfer (DKT) (Patacchiola et al., 2020) is a meta-learning method for fitting a deep kernel to a distribution of datasets. DKT jointly fits both the feature extractor parameters ϕ and base kernel parameters θ by maximising the expected GP marginal likelihood over a distribution of datasets (i.e., a meta-dataset).

To mitigate the overfitting issue of DKL using meta-learning, DKT makes a very strong assumption that different tasks in the task distribution are drawn from an identical GP prior over functions. Explicitly, this means that the data generating process is assumed to have the same noise level, same amplitude, and same "characteristic lengthscale" for every task in the

meta-dataset, which is a very restrictive assumption violated by most real-world problems. Taking drug discovery as an example, the drug properties in different datasets in a meta-dataset may have

- highly varying noise levels, so modelling all tasks with the same amount of observation noise will not be realistic;
- different output ranges and units for regression: for example, one task might have data in the range 1-20 μM, while another might have 0-100% inhibition, meaning that a single signal variance (kernel amplitude) will not model the data well;
- different "characteristic lengthscales": for some tasks, structurally similar molecules have very strongly correlated output labels, while for other tasks it is much weaker (i.e., there is much more variation in the labels of very similar molecules), suggesting that the "characteristic lengthscale" will be different.

Inevitably, trying to fit such a mis-specified model will result in a set of compromised base kernel parameters θ which fit all datasets fine on average but do not fit each individual dataset very well. This is the underfitting issue of DKT.

3.3 Adaptive Deep Kernel Fitting with Implicit Function Theorem

3.3.1 The ADKF-IFT Framework for Training Deep Kernel GPs

Let Θ and Φ respectively be the sets of base kernel and feature extractor parameters for a deep kernel GP. Denote the set of all parameters by $\Psi \equiv \Theta \times \Phi$. The key idea of the general ADKF-IFT framework is to partition the parameters into two disjoint subsets $\Psi = \Psi_{\text{adapt}} \times \Psi_{\text{meta}}$, where only a subset of the parameters $\psi_{\text{adapt}} \in \Psi_{\text{adapt}}$ will be adapted to each individual task by minimising a train loss $\mathcal{L}_T$, with the remaining set of parameters $\psi_{\text{meta}} \in \Psi_{\text{meta}}$ meta-learned during a *meta-training* phase to yield the best possible validation loss $\mathcal{L}_V$ *on average* over many related training tasks (*after* ψ_{meta} is separately adapted to each of these tasks). This can be naturally formalised as the following *bilevel optimisation* problem:

$$\psi^*_{\text{meta}} = \arg\min_{\psi_{\text{meta}}} \mathbb{E}_{p(\mathcal{T})}[\mathcal{L}_V(\psi_{\text{meta}}, \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}), \mathcal{T})], \tag{3.1}$$

$$\text{such that} \quad \psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T}) = \arg\min_{\psi_{\text{adapt}}} \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_\mathcal{T}). \tag{3.2}$$

Equation (3.1) and Equation (3.2) are most easily understood by separately considering the meta-learned parameters ψ_{meta} and the task-specific parameters ψ_{adapt}. For a given task $\mathcal{T}$ and

an arbitrary value for the meta-learned parameters ψ_{meta}, in Equation (3.2) the task-specific parameters ψ_{adapt} are chosen to minimise the *train loss* $\mathcal{L}_T$ evaluated on the task's support set $\mathcal{S}_\mathcal{T}$. That is, ψ_{adapt} is *adapted* to the support set $\mathcal{S}_\mathcal{T}$ of the task $\mathcal{T}$, with the aim of producing the best possible model on $\mathcal{S}_\mathcal{T}$ for the given value of ψ_{meta}. The result is a model with optimal task-specific parameters $\psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_\mathcal{T})$ for the given meta-learned parameters ψ_{meta} and task $\mathcal{T}$. The remaining question is how to choose a value for the meta-learned parameters ψ_{meta}, knowing that ψ_{adapt} will be adapted separately to each task. In Equation (3.1), we propose to choose ψ_{meta} to minimise the expected *validation loss* $\mathcal{L}_V$ over a distribution of training tasks $p(\mathcal{T})$. There are two reasons for this choice as detailed below.

1. On any given task $\mathcal{T}$, the validation loss usually reflects the performance metric of interest on the query set $\mathcal{Q}_\mathcal{T}$ of $\mathcal{T}$ (e.g., the prediction error).

2. Since the same value of ψ_{meta} will be used for all tasks, it makes sense to choose a value whose *expected performance* is good across many tasks drawn from $p(\mathcal{T})$. That is, ψ_{meta} is chosen such that a GP achieves the lowest possible average validation loss on the query set $\mathcal{Q}_\mathcal{T}$ of a random training task $\mathcal{T} \sim p(\mathcal{T})$ after ψ_{adapt} is adapted to the task's support set $\mathcal{S}_\mathcal{T}$.

In practice, ψ_{meta} would be optimised during a *meta-training* phase using a set of training tasks $\mathcal{D}$ to approximate Equation (3.1). After meta-training (i.e., at meta-test time), we make predictions for each unseen test task $\mathcal{T}_*$ using the joint GP posterior predictive distribution as defined in Equations (2.17) and (2.18) with optimal parameters ψ^*_{meta} and $\psi^*_{\text{adapt}}(\psi^*_{\text{meta}}, \mathcal{S}_{\mathcal{T}_*})$:

$$p(\mathcal{Q}^y_{\mathcal{T}_*} \mid \mathcal{Q}^x_{\mathcal{T}_*}, \mathcal{S}_{\mathcal{T}_*}, \psi^*_{\text{meta}}, \psi^*_{\text{adapt}}(\psi^*_{\text{meta}}, \mathcal{S}_{\mathcal{T}_*})). \tag{3.3}$$

Note that the description above does not specify a particular choice of $\Psi_{\text{meta}}, \Psi_{\text{adapt}}, \mathcal{L}_T, \mathcal{L}_V$. This is intentional, as there are many reasonable choices for these quantities. Because of this, we believe that ADKF-IFT should be considered a *general framework*, with a particular choice for these being an *instantiation* of the ADKF-IFT framework. We give examples of this in Section 3.3.3 and Section 3.3.4.

3.3.2 Efficient Meta-Training Algorithm

In general, optimising bilevel optimisation objectives such as Equation (3.1) is computationally complex, mainly because each evaluation of the objective requires solving a separate inner optimisation problem (3.2). Although calculating the *hypergradient* (i.e., total derivative) of the validation loss $\mathcal{L}_V$ with respect to the meta-learned parameters ψ_{meta} would allow

Algorithm 1 Exact hypergradient computation in ADKF-IFT

1: **Input:** a training task $\mathcal{T}'$ and the current meta-learned parameters ψ'_{meta}
2: Solve Equation (3.2) to obtain $\psi'_{\text{adapt}} = \psi^*_{\text{adapt}}(\psi'_{\text{meta}}, \mathcal{S}_{\mathcal{T}'})$
3: Compute $g_1 = \frac{\partial \mathcal{L}_V(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{T}')}{\partial \psi_{\text{meta}}}\Big|_{\psi'_{\text{meta}}, \psi'_{\text{adapt}}}$ and $g_2 = \frac{\partial \mathcal{L}_V(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{T}')}{\partial \psi_{\text{adapt}}}\Big|_{\psi'_{\text{meta}}, \psi'_{\text{adapt}}}$
4: Compute the Hessian $H = \frac{\partial^2 \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}'})}{\partial \psi_{\text{adapt}} \partial \psi_{\text{adapt}}^\top}\Big|_{\psi'_{\text{meta}}, \psi'_{\text{adapt}}}$
5: Solve the linear system $vH = g_2$ for v
6: Compute the mixed partial derivatives $P = \frac{\partial^2 \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}'})}{\partial \psi_{\text{adapt}} \partial \psi_{\text{meta}}^\top}\Big|_{\psi'_{\text{meta}}, \psi'_{\text{adapt}}}$
7: **Output:** the hypergradient $\frac{\mathrm{d}\mathcal{L}_V}{\mathrm{d}\psi_{\text{meta}}} = g_1 - vP$ ▷ Equations (3.4) and (3.5)

Equation (3.1) to be solved with gradient-based optimisation:

$$\frac{\mathrm{d}\,\mathcal{L}_V}{\mathrm{d}\psi_{\text{meta}}} = \frac{\partial\,\mathcal{L}_V}{\partial\psi_{\text{meta}}} + \frac{\partial\,\mathcal{L}_V}{\partial\psi^*_{\text{adapt}}}\frac{\partial\psi^*_{\text{adapt}}}{\partial\psi_{\text{meta}}}, \tag{3.4}$$

Equation (3.4) reveals that this requires calculating $\frac{\partial\psi^*_{\text{adapt}}}{\partial\psi_{\text{meta}}}$, i.e.,how the optimal task-specific parameters $\psi^*_{\text{adapt}}(\psi_{\text{meta}}, \mathcal{S}_{\mathcal{T}})$ change with respect to the meta-learned parameters ψ_{meta}. Calculating this naively with automatic differentiation platforms would require tracking the gradient through many iterations of the inner optimisation (3.2), which in practice requires too much memory to be feasible. Fortunately, because ψ^*_{adapt} is an optimum of the train loss $\mathcal{L}_T$, Cauchy's *Implicit Function Theorem* (IFT) provides a formula for calculating $\frac{\partial\psi^*_{\text{adapt}}}{\partial\psi_{\text{meta}}}$ for an arbitrary value of the meta-learned parameters ψ'_{meta} and a given task $\mathcal{T}'$:

$$\frac{\partial\psi^*_{\text{adapt}}}{\partial\psi_{\text{meta}}}\bigg|_{\psi'_{\text{meta}}} = -\left(\frac{\partial^2\,\mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}'})}{\partial\psi_{\text{adapt}}\partial\psi_{\text{adapt}}^\top}\right)^{-1}\frac{\partial^2\,\mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}'})}{\partial\psi_{\text{adapt}}\partial\psi_{\text{meta}}^\top}\bigg|_{\psi'_{\text{meta}}, \psi'_{\text{adapt}}}, \tag{3.5}$$

where $\psi'_{\text{adapt}} = \psi^*_{\text{adapt}}(\psi'_{\text{meta}}, \mathcal{S}_{\mathcal{T}'})$. A full statement of the implicit function theorem in the context of ADKF-IFT can be found in Appendix B.1. The only potential problem with Equation (3.5) is the computation and inversion of the Hessian matrix $\frac{\partial^2 \mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}})}{\partial \psi_{\text{adapt}} \partial \psi_{\text{adapt}}^\top}$. This computation can be done exactly if $|\Psi_{\text{adapt}}|$ is small, which is the case considered in this paper (as will be discussed in Section 3.3.4). Otherwise, an approximation to the inverse Hessian (e.g., Neumann approximation (Clarke et al., 2022; Lorraine et al., 2020)) could be used, which reduces both the memory and computational complexities to $\mathcal{O}(|\Psi|)$. Combining Equation (3.4) and Equation (3.5), we have a recipe for computing the hypergradient $\frac{\mathrm{d}\,\mathcal{L}_V}{\mathrm{d}\psi_{\text{meta}}}$ exactly for a single task, as summarised in Algorithm 1. The meta-learned parameters ψ_{meta} can then be updated with the expected hypergradient over $p(\mathcal{T})$.

3.3.3 ADKF-IFT as a Unification of Previous Methods

In prior work, the most common method used to train deep kernel GPs is to minimise the negative log marginal likelihood (NLML) on a single dataset (optionally with extra regularisation terms as will be discussed in Section 3.4.1). This is commonly referred to as *Deep Kernel Learning* (DKL) (Wilson et al., 2016b):

$$\psi^* = \arg\min_{\psi} \mathrm{NLML}\left(\psi, \mathcal{S}_{\mathcal{T}}\right). \tag{3.6}$$

The most notable departure from DKL is *Deep Kernel Transfer* (DKT) (Patacchiola et al., 2020), which instead proposes to train deep kernel GPs entirely using meta-learning, minimising the *expected* NLML over a distribution of training tasks:

$$\psi^* = \arg\min_{\psi} \mathbb{E}_{p(\mathcal{T})}[\mathrm{NLML}(\psi, \mathcal{T})]. \tag{3.7}$$

Interestingly, both DKL and DKT can be viewed as *special cases* of the general ADKF-IFT framework. It is easy to see that choosing the partition to be $\Psi_{\text{meta}} = \varnothing$, $\Psi_{\text{adapt}} = \Psi$ and the train loss $\mathcal{L}_T$ to be the NLML in Equation (3.1) and Equation (3.2) yields Equation (3.6): DKL is just ADKF-IFT if no parameters are meta-learned. Similarly, choosing the partition to be $\Psi_{\text{meta}} = \Psi$, $\Psi_{\text{adapt}} = \varnothing$ and the validation loss $\mathcal{L}_V$ to be the NLML in Equation (3.1) and Equation (3.2) yields Equation (3.7): DKT is just ADKF-IFT if all parameters are meta-learned. This makes ADKF-IFT *strictly more general* than these two previous methods.

3.3.4 Highlighted ADKF-IFT Instantiation

Among the many possible variations of ADKF-IFT, we would like to highlight the following instantiation:

- $\Psi_{\text{meta}} = \Phi$, i.e., all feature extractor parameters ϕ are meta-learned across tasks.
- $\Psi_{\text{adapt}} = \Theta$, i.e., all base kernel parameters θ (e.g., noise variance, lengthscales, amplitude) are adapted to each task.
- The train loss $\mathcal{L}_T$ is chosen to be the negative log GP marginal likelihood evaluated on the support set $\mathcal{S}_{\mathcal{T}}$, as is common practice for choosing GP base kernel parameters:

$$\begin{aligned}\mathcal{L}_T(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{S}_{\mathcal{T}}) &= -\log p(\mathcal{S}_{\mathcal{T}}^y \mid \mathcal{S}_{\mathcal{T}}^x, \psi_{\text{meta}}, \psi_{\text{adapt}}) \\ &= \frac{1}{2}(\mathcal{S}_{\mathcal{T}}^y)^\top K_{\mathcal{S}_{\mathcal{T}}}^{-1} \mathcal{S}_{\mathcal{T}}^y + \frac{1}{2}\log\det(K_{\mathcal{S}_{\mathcal{T}}}) + \frac{N_{\mathcal{S}_{\mathcal{T}}}}{2}\log(2\pi),\end{aligned} \tag{3.8}$$

where $K_{\mathcal{S}_{\mathcal{T}}} \coloneqq k_{\psi_{\text{meta}},\psi_{\text{adapt}}}(\mathcal{S}_{\mathcal{T}}^x, \mathcal{S}_{\mathcal{T}}^x) + \sigma^2 I_{N_{\mathcal{S}_{\mathcal{T}}}}$.

- The validation loss $\mathcal{L}_V$ is chosen to be the negative log joint GP predictive posterior evaluated on the query set $\mathcal{Q}_\mathcal{T}$ given the support set $\mathcal{S}_\mathcal{T}$, also due to its common usage for making predictions with GPs:

$$\begin{aligned}\mathcal{L}_V(\psi_{\text{meta}}, \psi_{\text{adapt}}, \mathcal{T}) &= -\log p(\mathcal{Q}_\mathcal{T}^y \mid \mathcal{Q}_\mathcal{T}^x, \mathcal{S}_\mathcal{T}, \psi_{\text{meta}}, \psi_{\text{adapt}}) \\ &= -\log \mathcal{N}(\mathcal{Q}_\mathcal{T}^y \mid K_{\mathcal{Q}_\mathcal{T}\mathcal{S}_\mathcal{T}} K_{\mathcal{S}_\mathcal{T}}^{-1} \mathcal{S}_\mathcal{T}^y, K_{\mathcal{Q}_\mathcal{T}} - K_{\mathcal{S}_\mathcal{T}\mathcal{Q}_\mathcal{T}}^\top K_{\mathcal{S}_\mathcal{T}}^{-1} K_{\mathcal{S}_\mathcal{T}\mathcal{Q}_\mathcal{T}}),\end{aligned} \tag{3.9}$$

  where $K_{\mathcal{Q}_\mathcal{T}} \coloneqq k_{\psi_{\text{meta}},\psi_{\text{adapt}}}(\mathcal{Q}_\mathcal{T}^x, \mathcal{Q}_\mathcal{T}^x) + \sigma^2 I_{N_{\mathcal{Q}_\mathcal{T}}}$ and $K_{\mathcal{S}_\mathcal{T}\mathcal{Q}_\mathcal{T}} \coloneqq k_{\psi_{\text{meta}},\psi_{\text{adapt}}}(\mathcal{S}_\mathcal{T}^x, \mathcal{Q}_\mathcal{T}^x)$.

There are several benefits to this choice as detailed below.

1. This particular choice of loss functions has the advantage that the prediction procedure during meta-testing as defined in Equation (3.3) exactly matches the meta-training procedure, thereby closely following the principle of *learning to learn*.
2. The partition of parameters can be intuitively understood as meta-learning a generally useful feature extractor f_ϕ such that it is possible on average to fit a low-loss GP to the feature representations extracted by f_ϕ for each individual task. This is very similar to previous transfer learning approaches.
3. Since most GP base kernels have only a handful of parameters, the Hessian in Equation (3.5) can be computed and inverted exactly during meta-training using Algorithm 1; this removes any need for Hessian approximations.
4. The inner optimisation (3.2) for ψ_{adapt} is computationally efficient, as it does not require backpropagating through the feature extractor f_ϕ.

More generally, ADKF-IFT combines DKL and DKT in a way that can potentially inherit the strengths of both methods and the weaknesses of neither. By adapting the base kernel parameters θ specifically to each task, it prevents underfitting due to varying ranges, lengthscales, or noise levels between datasets. By meta-learning the feature extractor on many datasets, it prevents overfitting as observed by Ober et al. (2021). This advantage is both *theoretically principled* (by solving a bilevel optimisation objective using the implicit function theorem) and *empirically observable* (as will be shown in Section 3.5).

The relationship between these methods are visualised in Figure 3.1. Panel (c) shows DKL, which trains a separate deep kernel GP for each task. It is not hard to imagine that this can lead to severe overfitting for small datasets, which has been observed empirically by Ober et al. (2021). Panel (b) shows DKT, which prevents overfitting by fitting one deep kernel GP for all tasks. However, this implicitly makes a strong assumption that all tasks come from an identical distribution over functions, including the same noise level, same amplitude, and same

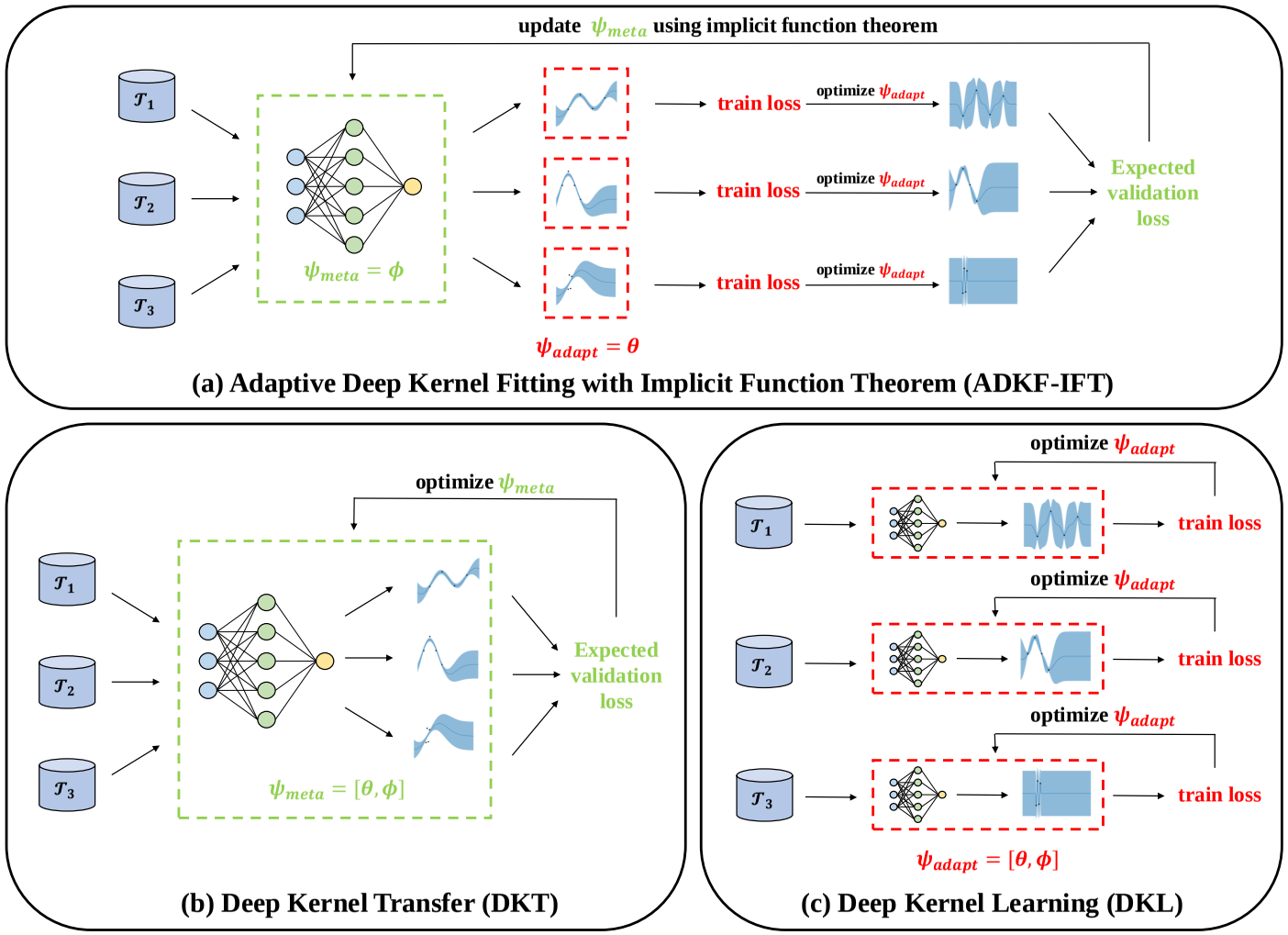


Figure 3.1 A contrastive diagram illustrating the training procedures of ADKF-IFT, DKT and DKL. (a) ADKF-IFT meta-learns the feature extractor parameters ϕ across all tasks and adapts the base kernel parameters θ to each task under a bilevel optimisation framework. (b) DKT meta-learns all parameters across all tasks. (c) DKL adapts all parameters to each task.

characteristic lengthscales, which is unlikely to hold in practice. Panel (a) shows ADKF-IFT, which allows these important parameters to be adapted, while still regularising the feature extractor with meta-learning. We conjecture that adapting the base kernel parameters is more appropriate given the expected differences between tasks: two related tasks are more likely to have different noise levels or characteristic lengthscales than to require substantially different feature representations (as discussed in Section 3.2.2).

3.4 Related Work

3.4.1 Deep Kernel GPs

ADKF-IFT is part of a growing body of literature of techniques to train deep kernel GPs. As discussed in Section 3.3.3, ADKF-IFT *generalises* DKL (Wilson et al., 2016b) and DKT (Patacchiola et al., 2020), which exclusively use single-task learning and meta-learning, respectively. Liu et al. (2020) and van Amersfoort et al. (2021) propose adding regularisation terms to the loss of DKL in order to mitigate overfitting. These works are better viewed as *complementary* to ADKF-IFT rather than *alternatives*: their respective regularisation terms could easily be added to $\mathcal{L}_T$ in Equation (3.2) to improve performance. However, the regularisation strategies in both of these papers are designed for continuous inputs only, limiting their applicability to structured data like molecules.

Interestingly, the preprint of Tossou et al. (2019) proposes an alternative way to meta-train adaptive deep kernel GPs called ADKL-GP, where adaptation is performed by conditioning the feature extractor on an embedding of the entire support set rather than adjusting a subset of the kernel parameters as in ADKF-IFT. In general, their empirical results were not very strong, and in our opinion the method is very prone to overfitting. This is because the training objective for ADKL-GP is equivalent to the objective for DKT with an added contrastive loss, weighted by a sensitive hyperparameter γ (see Equation (13) of Tossou et al. (2019)), which can be interpreted as balancing the degree of regularisation between two extremes:

- If $\gamma = 0$, there is no regularisation of the task encoding network, making significant overfitting to the meta-dataset possible. This is effectively equivalent to standard DKL.
- As $\gamma \to \infty$, the regularisation becomes infinitely strong, causing the task embeddings $z_\mathcal{T}$ to collapse, and thereby preventing them from transmitting any information about specific datasets. With no information from $z_\mathcal{T}$ in this case, the objective is essentially the same as that of DKT.

For this method to be useful it would appear that γ would need to be carefully tuned to balance between these extremes. Tossou et al. (2019) perform a grid search over all hyperparameters

including $\gamma \in \{0, 0.01, 0.1\}$ but find no consistent trend besides $\gamma > 0$ being slightly helpful, although the differences in performance were small. This suggests that the method may be difficult to use in practice. ADKF-IFT however has no such tunable hyperparameters, which we view as a significant strength. Instead, the balance between DKL and DKT is controlled by selecting which parameters are adapted and meta-learned, which is much more interpretable and makes it easier to use in practice.

3.4.2 Meta-Learning

ADKF-IFT can also be viewed as a meta-learning algorithm comparable to many previously-proposed methods (Chen et al., 2021, 2020; Garnelo et al., 2018; Lake et al., 2011; Liu et al., 2021; Park and Oliva, 2019; Patacchiola et al., 2022; Tian et al., 2020; Triantafillou et al., 2020; Vinyals et al., 2016; Wistuba and Grabocka, 2021). One distinguishing feature of ADKF-IFT is that it is specially designed for deep kernel GPs, whereas most methods from computer vision are designed exclusively for neural network models, which as previously stated are unsuitable when reliable uncertainty estimates are required. Furthermore, many of these algorithms such as ProtoNet (Snell et al., 2017) are designed principally or exclusively for classification, while ADKF-IFT is suited to both regression and classification. Compared to model-agnostic frameworks like MAML (Finn et al., 2017; Rajeswaran et al., 2019) which meta-learn a common initialisation for adaptation of all parameters, ADKF-IFT meta-learns a subset of parameters across tasks and adapts the rest of parameters to each task.

Modular meta-learning with shrinkage (Chen et al., 2020) shares some similarities with ADKF-IFT: at a high level it also divides model parameters into meta-learned and adapted parameters, and optimises the meta-learned parameters using the gradient of the validation loss after the adapted parameters have been adjusted to minimise the training loss using IFT. However, there are three main differences between Chen et al. (2020) and ADKF-IFT:

1. Chen et al. (2020) consider a model where the meta-learned parameters ϕ are the means and variances of a Gaussian prior over model parameters, whereas ϕ is a subset of the parameters of a deep kernel GP in ADKF-IFT.

2. Chen et al. (2020) consider the case where there are too many adapted parameters θ to form the exact Hessian in IFT. They instead use a conjugate gradient approximation. Although some instantiations of ADKF-IFT could may this, in our highlighted version the Hessian can be computed exactly, which we view as a significant advantage.

3. The goal of the method in Chen et al. (2020) is to learn which parameters should be meta-learned, while in ADKF-IFT this must be pre-specified (and we give guidance for doing so in a way that results in transferable meta-learned features).

3.4.3 Multi-Task GPs

ADKF-IFT can be considered as a method to learn multi-task GPs. The dominant approach to this problem in prior works is to learn a shared kernel for all data points across all tasks, transmitting information by explicitly modelling the covariance between data from different tasks (Bonilla et al., 2007; Forrester et al., 2007; Kennedy and O'Hagan, 2000; Poloczek et al., 2017; Swersky et al., 2013). The main difference between methods in this family is the exact form of the kernel (Tighineanu et al., 2022), which is typically assumed to have a particular structure (e.g., Kronecker product or weighted sum of simpler kernels). ADKF-IFT cannot be naturally viewed in this way because the covariance between data points from separate tasks is always zero; information is instead transmitted across tasks via a shared set of model parameters. Therefore, we believe that ADKF-IFT is a significant departure from the dominant paradigm in GP transfer learning.

3.4.4 Implicit Function Theorem in Machine Learning

The IFT employed in our work has been used in many previous machine learning papers in various contexts, e.g., neural architecture search (Zhang et al., 2021), hyperparameter-tuning (Bengio, 2000; Clarke et al., 2022; Lorraine et al., 2020; Luketina et al., 2016; Pedregosa, 2016), and meta-learning (Chen et al., 2020; Lee et al., 2019; Rajeswaran et al., 2019).

One challenge of applying IFT is the computation of the inverse Hessian. While earlier work compute this explicitly (Bengio, 2000; Larsen et al., 1996) as in ADKF-IFT, recent work employ various approximation techniques such as Neumann approximation (Clarke et al., 2022; Lorraine et al., 2020), conjugate gradient (Chen et al., 2020; Pedregosa, 2016; Rajeswaran et al., 2019; Wang et al., 2020), and the identity matrix (Luketina et al., 2016) to retain computational tractability.

3.5 Empirical Evaluation

This section presents empirical evaluation results for the performance of the specific instantiation of ADKF-IFT from Section 3.3.4. We choose to focus our experiments exclusively on molecular property prediction and optimisation tasks because we believe that this application would benefit greatly from better GP models: firstly because many existing methods struggle on small datasets of size $\sim 10^2$ which are ubiquitous in chemistry, and secondly because many tasks in chemistry require high-quality uncertainty estimates. First, we evaluate ADKF-IFT on four commonly used benchmark tasks from MoleculeNet (Wu et al., 2018), finding that ADKF-IFT achieves state-of-the-art results on most tasks (Section 3.5.1). Second, we evaluate ADKF-IFT on the larger-scale FS-Mol benchmark (Stanley et al., 2021), finding that

Table 3.1 Statistics of four few-shot molecular property prediction benchmarks from MoleculeNet.

Statistic	MoleculeNet benchmark task			
	Tox21	SIDER	MUV	ToxCast
#compounds	8,014	1,427	93,127	8,615
#tasks	12	27	17	617
#training tasks	9	21	12	450
#test tasks	3	6	5	167

ADKF-IFT is the best-performing method (Section 3.5.2). In particular, our results support the hypothesis from Section 3.3.4 that ADKF-IFT achieves a better balance between overfitting and underfitting than DKL and DKT. Finally, we show that the ADKF-IFT feature representation is transferable to out-of-domain (OOD) molecular property prediction and optimisation tasks (Section 3.5.3). The detailed configurations of ADKF-IFT for all experiments considered in this section are shown in Appendix B.2.

3.5.1 Few-shot Molecular Property Prediction on MoleculeNet

Benchmark and Baselines

We compare **ADKF-IFT** with two types of baselines on four few-shot molecular property classification benchmark tasks (Tox21, SIDER, MUV, and ToxCast) from MoleculeNet (Wu et al., 2018). The statistics of these benchmark tasks are summarised in Table 3.1.

The first type of baseline methods train their feature extractors from scratch: **Siamese** (Koch, 2015), **ProtoNet** (Snell et al., 2017), **MAML** (Finn et al., 2017), **TPN** (Liu et al., 2019b), **ELGNN** (Kim et al., 2019), **IterRefLSTM** (Altae-Tran et al., 2017) and **PAR** (Wang et al., 2021). The second type of baseline methods fine-tune a pretrained feature extractor: **Pre-GNN** (Hu et al., 2020), **Meta-MGNN** (Guo et al., 2021) and **Pre-PAR** (Wang et al., 2021). **Pre-ADKF-IFT** refers to ADKF-IFT starting from a pretrained feature extractor. All compared methods in this section use GIN (Xu et al., 2019) as their feature extractors. The pretrained weights for the methods of the second type are provided by Hu et al. (2020).

Evaluation Procedure

We follow exactly the same evaluation procedure as that in Guo et al. (2021); Hu et al. (2020); Wang et al. (2021). The task-level metric is AUROC (area under the receiver operating characteristic curve). We report the averaged performance over ten runs with different random seeds for each compared method at the support set size 20 (i.e., 2-way 10-shot, as the support

Table 3.2 Mean test performance (AUROC%) with standard deviations of all compared methods on MoleculeNet benchmark tasks at support set size 20 (i.e., 2-way 10-shot).

Method	MoleculeNet benchmark task (#compounds)			
	Tox21 (8,014)	SIDER (1,427)	MUV (93,127)	ToxCast (8,615)
Siamese	80.40 ± 0.35	71.10 ± 4.32	59.59 ± 5.13	-
ProtoNet	74.98 ± 0.32	64.54 ± 0.89	65.88 ± 4.11	63.70 ± 1.26
MAML	80.21 ± 0.24	70.43 ± 0.76	63.90 ± 2.28	66.79 ± 0.85
TPN	76.05 ± 0.24	67.84 ± 0.95	65.22 ± 5.82	62.74 ± 1.45
ELGNN	81.21 ± 0.16	72.87 ± 0.73	65.20 ± 2.08	63.65 ± 1.57
IterRefLSTM	81.10 ± 0.17	69.63 ± 0.31	45.56 ± 5.12	-
PAR	82.06 ± 0.12	$\mathbf{74.68 \pm 0.31}$	66.48 ± 2.12	69.72 ± 1.63
ADKF-IFT	$\mathbf{82.43 \pm 0.60}$	67.72 ± 1.21	$\mathbf{98.18 \pm 3.05}$	$\mathbf{72.07 \pm 0.81}$
Pre-GNN	82.14 ± 0.08	73.96 ± 0.08	67.14 ± 1.58	73.68 ± 0.74
Meta-MGNN	82.97 ± 0.10	75.43 ± 0.21	68.99 ± 1.84	-
Pre-PAR	84.93 ± 0.11	$\mathbf{78.08 \pm 0.16}$	69.96 ± 1.37	75.12 ± 0.84
Pre-ADKF-IFT	$\mathbf{86.06 \pm 0.35}$	70.95 ± 0.60	$\mathbf{95.74 \pm 0.37}$	$\mathbf{76.22 \pm 0.13}$

sets in MoleculeNet are balanced). We did not perform 1-shot learning, as it is an unrealistic setting in real-world drug discovery tasks. All baseline results are taken from Wang et al. (2021).

Performance

Table 3.2 shows that ADKF-IFT and Pre-ADKF-IFT achieve the best performance on Tox21, MUV, and ToxCast. In general, the larger the dataset is, the larger the performance gains of our method over other baselines are, highlighting the scalability of our method. In particular, our method outperforms all baselines by a wide margin on MUV due to the relatively large amount of available compounds, but underperforms many baselines on SIDER due to a lack of compounds.

Note that our method achieves near 100% AUROC on MUV. One possible cause is that the MUV dataset is extremely imbalanced with the positively labelled compounds tending to be very similar to each other (Wang et al., 2024). This is because MoleculeNet is an old benchmark with many known issues (Walters, 2023), including data leakage in the train/test splits and presence of duplicate compounds. As a result, methods like ADKF-IFT which perform nearest-neighbour comparison to examples in the support set could perform abnormally well.

3.5.2 Few-shot Molecular Property Prediction on FS-Mol

Benchmark

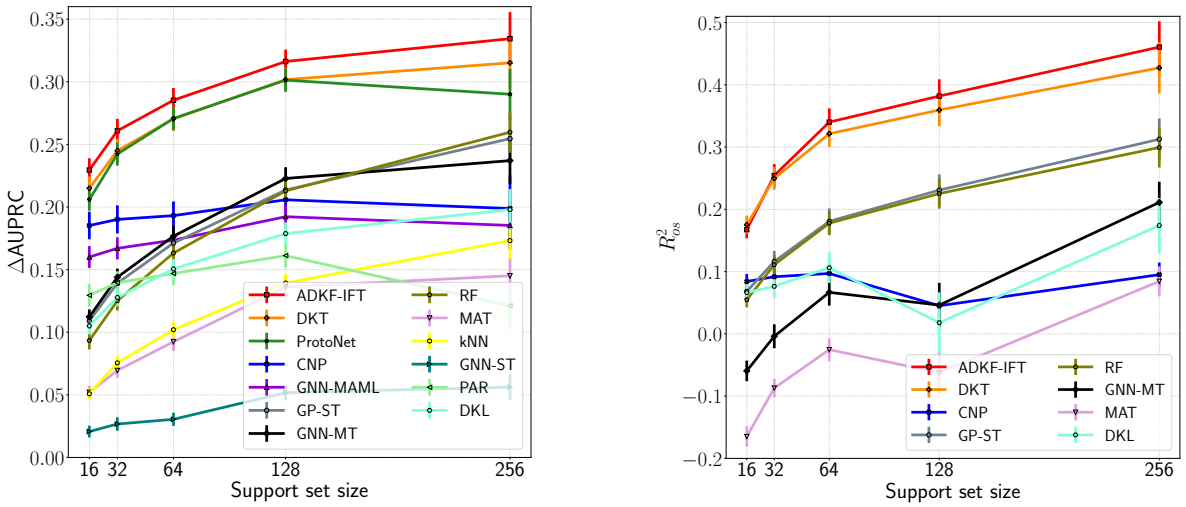
We further conduct evaluation on the FS-Mol benchmark (Stanley et al., 2021), which contains a carefully constructed set of few-shot learning tasks for molecular property prediction. FS-Mol contains over 5,000 tasks with 233,786 unique compounds from ChEMBL27 (Mendez et al., 2019), split into training (4,938 tasks), validation (40 tasks), and test (157 tasks) sets. Each task is associated with a protein target. The original benchmark only considers binary classification of active/inactive compounds, but we additionally include the regression task for the actual numeric activity target IC50 or EC50 in our evaluation as well, as it is a desired and more preferred task to do in real-world drug discovery projects.

Baselines

We compare **ADKF-IFT** with four categories of baselines.

1. *Single-task learning methods*: random forest (**RF**), k-nearest neighbours (**kNN**), single-task GP with Tanimoto kernel (**GP-ST**) (Ralaivola et al., 2005), single-task GNN (**GNN-ST**) (Gilmer et al., 2017), deep kernel learning (**DKL**) (Wilson et al., 2016b).
2. *Multi-task pretraining method*: multi-task GNN (**GNN-MT**) (Corso et al., 2020; Gilmer et al., 2017).
3. *Self-supervised pretraining method*: molecule attention transformer (**MAT**) (Maziarka et al., 2020).
4. *Meta-learning methods*: property-aware relation networks (**PAR**) (Wang et al., 2021), prototypical network with Mahalanobis distance (**ProtoNet**) (Snell et al., 2017), model-agnostic meta-learning (**GNN-MAML**) (Finn et al., 2017), conditional neural process (**CNP**) (Garnelo et al., 2018), deep kernel transfer (**DKT**) (Patacchiola et al., 2020).

The GNN feature extractor architecture f_ϕ used for DKL, PAR, CNP, DKT, and ADKF-IFT is the same as that used for ProtoNet, GNN-ST, GNN-MT, and GNN-MAML in Stanley et al. (2021). All multi-task and meta-learning methods are trained from scratch on FS-Mol training tasks. MAT is pretrained on 2 million molecules sampled from the ZINC15 dataset (Sterling and Irwin, 2015). The classification results for RF, kNN, GNN-ST, GNN-MT, MAT, ProtoNet, and GNN-MAML are reproduced according to Stanley et al. (2021). Detailed configurations of all baselines can be found in Appendix B.3.



(a) Classification (157 tasks)

(b) Regression (111 tasks)

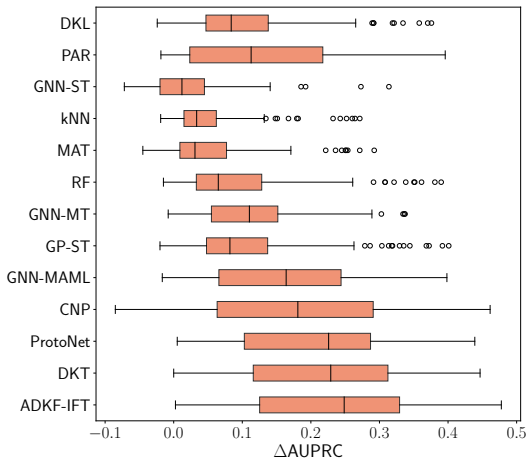
Figure 3.2 Mean performance with standard errors of all compared methods on all FS-Mol test tasks.

Evaluation Procedure

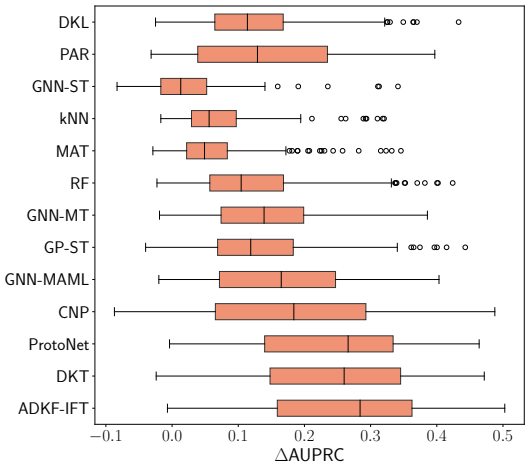
The task-level metrics for binary classification and regression are ΔAUPRC (change in area under the precision-recall curve) and R^2_{os} (predictive/out-of-sample coefficient of determination), respectively. We follow exactly the same evaluation procedure as that in Stanley et al. (2021), where the averaged performance over ten different stratified support/query random splits of every test task is reported for each compared method. This evaluation process is performed for five different support set sizes 16, 32, 64, 128, and 256. Note that unlike MoleculeNet, the support sets are generally unbalanced for the classification task in FS-Mol, which is natural as the majority of the candidate molecules are inactive in drug discovery.

Overall Performance

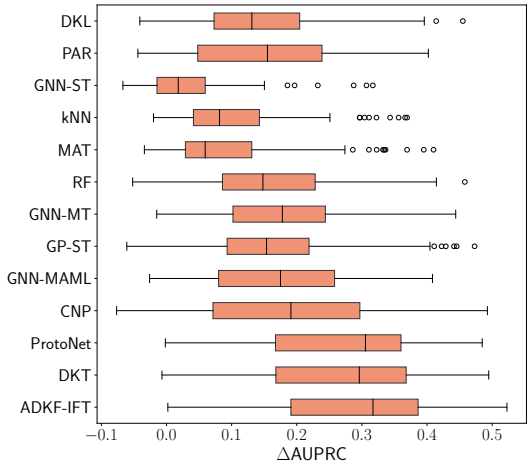
Figure 3.2 shows the overall test performance of all compared methods aggregated over all tasks. Note that RF is a strong baseline method, as it is widely used in real-world drug discovery projects and has comparable performance to many pretraining methods. The results indicate that ADKF-IFT outperforms all the other compared methods at all considered support set sizes for the classification task. For the regression task, the performance gains of ADKF-IFT over the second best method, namely DKT, get larger as the support set size increases. Figure 3.3 and Figure 3.4 show the box plots for the classification and regression performances of all compared methods on all FS-Mol test tasks, respectively. These plots are a disaggregated representation of the results in Figure 3.2.



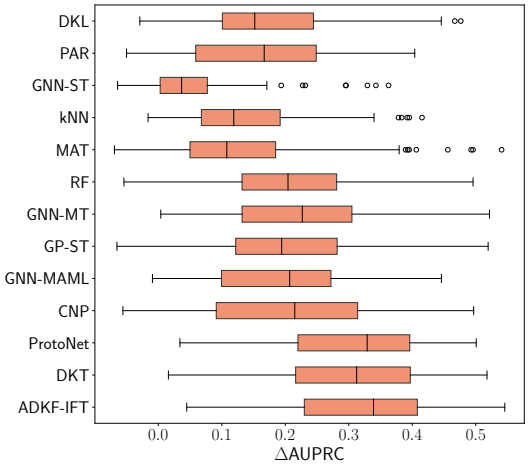
(a) $N_{\mathcal{S}_{\mathcal{T}_*}} = 16$



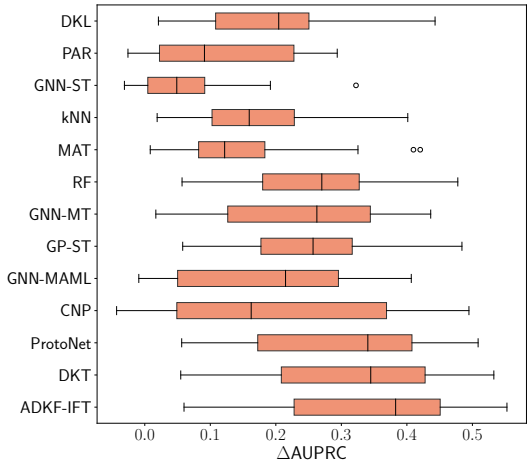
(b) $N_{\mathcal{S}_{\mathcal{T}_*}} = 32$



(c) $N_{\mathcal{S}_{\mathcal{T}_*}} = 64$

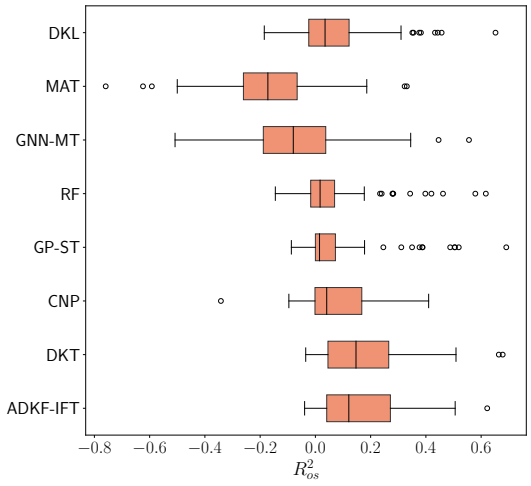


(d) $N_{\mathcal{S}_{\mathcal{T}_*}} = 128$

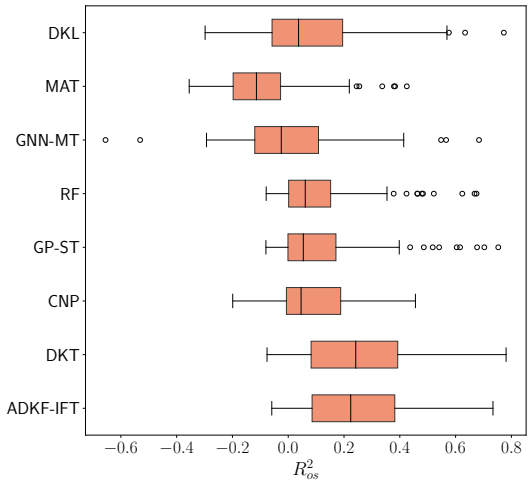


(e) $N_{\mathcal{S}_{\mathcal{T}_*}} = 256$

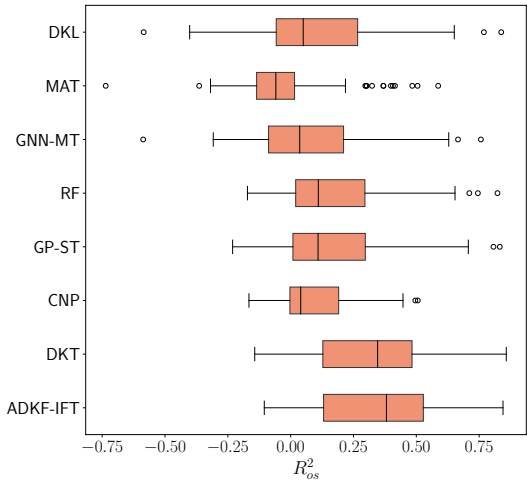
Figure 3.3 Box plots for the classification performance of all compared methods on 157 FS-Mol test tasks at different support set sizes.



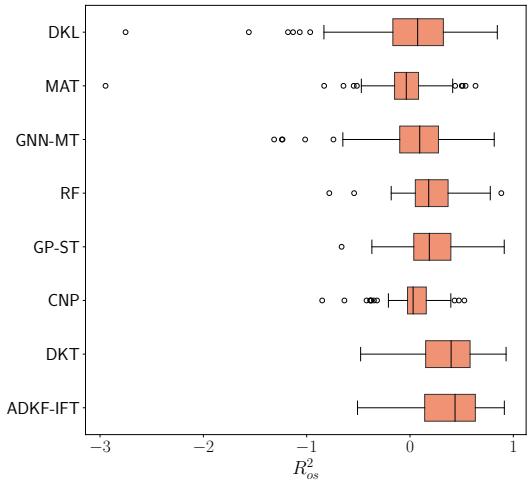
(a) $N_{\mathcal{S}_{\mathcal{T}_*}} = 16$



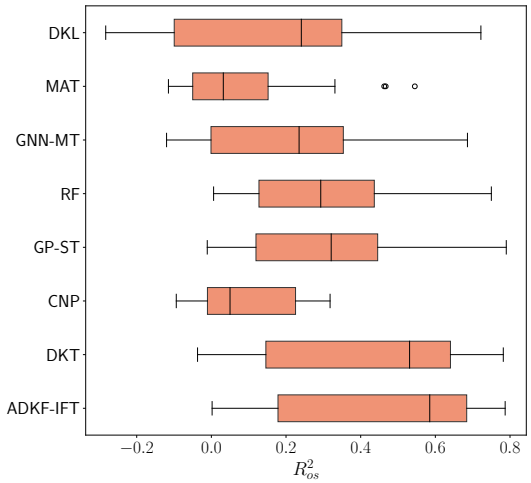
(b) $N_{\mathcal{S}_{\mathcal{T}_*}} = 32$



(c) $N_{\mathcal{S}_{\mathcal{T}_*}} = 64$



(d) $N_{\mathcal{S}_{\mathcal{T}_*}} = 128$



(e) $N_{\mathcal{S}_{\mathcal{T}_*}} = 256$

Figure 3.4 Box plots for the regression performance of all compared methods on 111 FS-Mol test tasks at different support set sizes.

Table 3.3 Mean rank of performance for all compared methods on all FS-Mol test tasks.

(a) Classification (157 tasks)

Method	Support set size				
	16	32	64	128	256
GNN-ST	11.29	11.53	11.75	11.85	12.19
kNN	10.89	10.48	10.33	10.15	9.37
MAT	10.43	10.44	10.19	9.69	9.70
RF	8.15	7.89	7.06	6.25	4.47
PAR	7.70	7.98	8.30	8.83	10.81
GNN-MT	7.33	7.18	7.08	6.59	6.53
DKL	7.28	7.49	7.98	8.42	8.21
GP-ST	6.71	6.57	6.28	6.18	5.14
GNN-MAML	6.36	6.92	7.42	7.89	8.90
CNP	5.00	5.81	6.36	6.91	7.78
ProtoNet	4.00	3.40	3.11	2.98	3.85
DKT	3.44	3.19	2.99	2.99	2.67
ADKF-IFT	**2.41**	**2.12**	**2.14**	**2.26**	**1.38**

(b) Regression (111 tasks)

Method	Support set size				
	16	32	64	128	256
MAT	7.60	7.45	7.26	7.06	7.19
GNN-MT	6.61	6.40	6.15	5.95	5.58
RF	5.00	4.47	4.16	3.72	3.56
DKL	4.42	5.16	5.63	6.10	6.35
GP-ST	4.23	4.14	3.87	3.37	3.07
CNP	3.88	4.45	4.95	5.73	6.47
DKT	**2.12**	2.08	2.29	2.32	2.43
ADKF-IFT	**2.12**	**1.86**	**1.68**	**1.74**	**1.36**

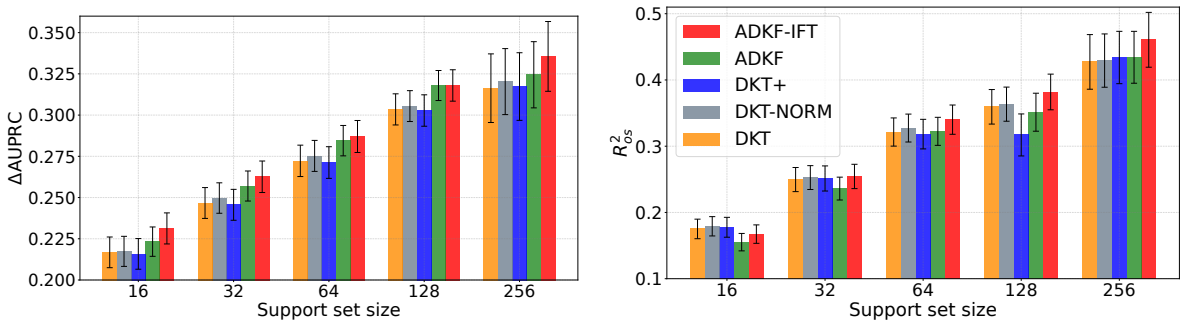
Table 3.3 shows that ADKF-IFT achieves the best mean rank for both classification and regression tasks at all considered support set sizes. The trends of these mean ranks are consistent to the aggregated performance shown in Figure 3.2.

Ablation Study

We perform ablation study to verify the following two hypotheses.

1. The bilevel optimisation objective for ADKF-IFT is essential for learning informative feature representations.
2. The performance gains of ADKF-IFT are not simply caused by tuning the base kernel parameters θ at meta-test time.

We consider three ablation models: DKT-NORM, DKT+ and ADKF. The test performance of these models are shown in Figure 3.5. For ADKF, we follow the ADKF-IFT training scheme but assume $\frac{\partial \theta^*}{\partial \phi} = 0$, i.e., updating the feature extractor parameters ϕ with the *direct gradient* $\frac{\partial \mathcal{L}_V}{\partial \phi}$ rather than the hypergradient $\frac{\mathrm{d}\mathcal{L}_V}{\mathrm{d}\phi}$. The results show that ADKF consistently underperforms ADKF-IFT, indicating that the hypergradient for the bilevel optimisation objective has non-negligible contributions to learning better feature representations. For DKT+, we take a model trained by DKT and adapt the base kernel parameters θ on each task at meta-test time. For DKT-NORM, the neural network features and labels are normalised for each task. The results show that DKT+ and DKT-NORM do not improve upon DKT, indicating that tuning the base kernel parameters at meta-test time or normalising the features and labels is not sufficient for obtaining better test performance with DKT.



(a) Classification (157 tasks)

(b) Regression (111 tasks)

Figure 3.5 Mean performance with standard errors of ablation models on all FS-Mol test tasks. ADKF is like ADKF-IFT but assuming $\frac{\partial \theta^*}{\partial \phi} = 0$, i.e., updating ϕ with the direct gradient $\frac{\partial \mathcal{L}_V}{\partial \phi}$. DKT+ is like DKT but tuning the base kernel parameters θ during meta-testing. DKT-NORM is like DKT but with normalised neural network features and labels.

Additionally, Appendix B.4 provides visualisations of the distributions of the optimal ADKF-IFT base kernel parameters against the DKT optimal base kernel parameters, confirming that ADKF-IFT learns to make more informative predictions than DKT.

Statistical Comparison

We perform two-sided Wilcoxon signed-rank tests (Wilcoxon, 1992) to compare the performance of ADKF-IFT and the next best method, DKT, and two ablation models DKT+ and ADKF. The exact p-values from these statistical tests can be found in Table 3.4. The test results indicate that their median performance difference is non-zero (i.e., ADKF-IFT significantly outperforms these three methods) in the majority of cases.

Table 3.4 p-values from the two-sided Wilcoxon signed-rank test for statistical comparisons between ADKF-IFT and DKT/DKT+/ADKF. The null hypothesis is that the median of their performance differences on all FS-Mol test tasks is zero. The significance level is set to $\alpha = 0.05$.

Compared models	Task type	Support set size				
		16	32	64	128	256
ADKF-IFT vs DKT	Classification	$\mathbf{1.4 \times 10^{-12}}$	$\mathbf{8.1 \times 10^{-14}}$	$\mathbf{2.3 \times 10^{-12}}$	$\mathbf{1.0 \times 10^{-8}}$	$\mathbf{3.4 \times 10^{-7}}$
	Regression	8.2×10^{-2}	9.6×10^{-2}	$\mathbf{3.7 \times 10^{-5}}$	$\mathbf{7.1 \times 10^{-5}}$	$\mathbf{9.8 \times 10^{-7}}$
ADKF-IFT vs DKT+	Classification	$\mathbf{3.2 \times 10^{-13}}$	$\mathbf{7.0 \times 10^{-15}}$	$\mathbf{2.3 \times 10^{-13}}$	$\mathbf{1.2 \times 10^{-9}}$	$\mathbf{1.6 \times 10^{-6}}$
	Regression	$\mathbf{3.2 \times 10^{-2}}$	4.2×10^{-1}	$\mathbf{3.4 \times 10^{-5}}$	$\mathbf{5.2 \times 10^{-10}}$	$\mathbf{1.2 \times 10^{-5}}$
ADKF-IFT vs ADKF	Classification	$\mathbf{1.7 \times 10^{-2}}$	1.1×10^{-1}	4.8×10^{-1}	8.3×10^{-1}	$\mathbf{1.6 \times 10^{-3}}$
	Regression	$\mathbf{2.8 \times 10^{-3}}$	$\mathbf{4.2 \times 10^{-4}}$	$\mathbf{1.3 \times 10^{-3}}$	$\mathbf{4.1 \times 10^{-6}}$	$\mathbf{1.3 \times 10^{-5}}$

Table 3.5 Mean performance with standard errors of top performing methods on FS-Mol test tasks within each sub-benchmark (broken down by EC category) at support set size 64 (the median of all considered support sizes). Note that class 2 is most common in the FS-Mol training set ($\sim 1,500$ training tasks), whereas classes 6 and 7 are least common in the FS-Mol training set (< 50 training tasks each).

(a) Classification (ΔAUPRC)

FS-Mol sub-benchmark (EC category)			Method				
Class	Description	#tasks	RF	GP-ST	ProtoNet	DKT	ADKF-IFT
1	oxidoreductases	7	0.156 ± 0.044	0.152 ± 0.040	0.137 ± 0.037	0.145 ± 0.040	$\mathbf{0.160 \pm 0.045}$
2	kinases	125	0.152 ± 0.009	0.161 ± 0.009	0.285 ± 0.010	0.282 ± 0.010	$\mathbf{0.299 \pm 0.010}$
3	hydrolases	20	0.229 ± 0.032	0.230 ± 0.032	0.245 ± 0.034	0.254 ± 0.034	$\mathbf{0.262 \pm 0.033}$
4	lysases	2	0.276 ± 0.182	$\mathbf{0.284 \pm 0.189}$	0.265 ± 0.211	0.272 ± 0.206	0.279 ± 0.201
5	isomerases	1	0.166 ± 0.040	$\mathbf{0.212 \pm 0.052}$	0.172 ± 0.044	0.204 ± 0.058	0.198 ± 0.046
6	ligases	1	0.149 ± 0.035	0.199 ± 0.028	0.170 ± 0.028	0.229 ± 0.013	$\mathbf{0.231 \pm 0.022}$
7	translocases	1	$\mathbf{0.128 \pm 0.039}$	0.109 ± 0.049	0.099 ± 0.028	0.122 ± 0.022	0.109 ± 0.033
	all enzymes	157	0.163 ± 0.009	0.171 ± 0.009	0.271 ± 0.009	0.271 ± 0.010	$\mathbf{0.285 \pm 0.010}$

(b) Regression (R^2_{os})

FS-Mol sub-benchmark (EC category)			Method				
Class	Description	#tasks	RF	GP-ST	CNP	DKT	ADKF-IFT
1	oxidoreductases	6	0.108 ± 0.087	0.103 ± 0.076	-0.012 ± 0.011	0.098 ± 0.078	$\mathbf{0.116 \pm 0.079}$
2	kinases	82	0.160 ± 0.019	0.162 ± 0.022	0.127 ± 0.017	0.343 ± 0.022	$\mathbf{0.363 \pm 0.024}$
3	hydrolases	19	0.256 ± 0.058	0.267 ± 0.061	0.014 ± 0.015	0.295 ± 0.063	$\mathbf{0.310 \pm 0.062}$
4	lysases	2	0.418 ± 0.405	0.417 ± 0.416	0.100 ± 0.068	0.440 ± 0.418	$\mathbf{0.442 \pm 0.403}$
5	isomerases	1	0.125 ± 0.077	0.086 ± 0.082	-0.012 ± 0.010	0.209 ± 0.113	$\mathbf{0.226 \pm 0.063}$
6	ligases	1	0.182 ± 0.040	0.202 ± 0.079	0.002 ± 0.004	0.277 ± 0.035	$\mathbf{0.279 \pm 0.043}$
	all enzymes	111	0.178 ± 0.019	0.181 ± 0.021	0.097 ± 0.014	0.321 ± 0.021	$\mathbf{0.340 \pm 0.022}$

Table 3.6 Descriptions of four out-of-domain molecular design tasks. the datasets for the molecular docking and material design tasks are sub-sampled from the much larger datasets provided in DOCKSTRING (García-Ortegón et al., 2022) and Harvard Clean Energy Project (Hachmann et al., 2011), respectively. The datasets for the antibiotic discovery and antiviral drug design tasks are taken from the antibiotic training set and the COVID Moonshot dataset provided in Stokes et al. (2020) and Achdout et al. (2022), respectively.

Molecular design task	Data source	#compounds	Target	Target source
Docking (ESR2)	DOCKSTRING training set	2,312	binding score	AutoDock Vina
Antibiotic (E. coli BW25113)	Antibiotic training set	2,335	relative growth	screening
Antiviral (SARS-CoV-2)	COVID Moonshot	1,926	pIC50 Fluorescence	experimental lab
Material (OPV)	Harvard Clean Energy Project	2,012	power conversion efficiency	DFT

Sub-benchmark Performance

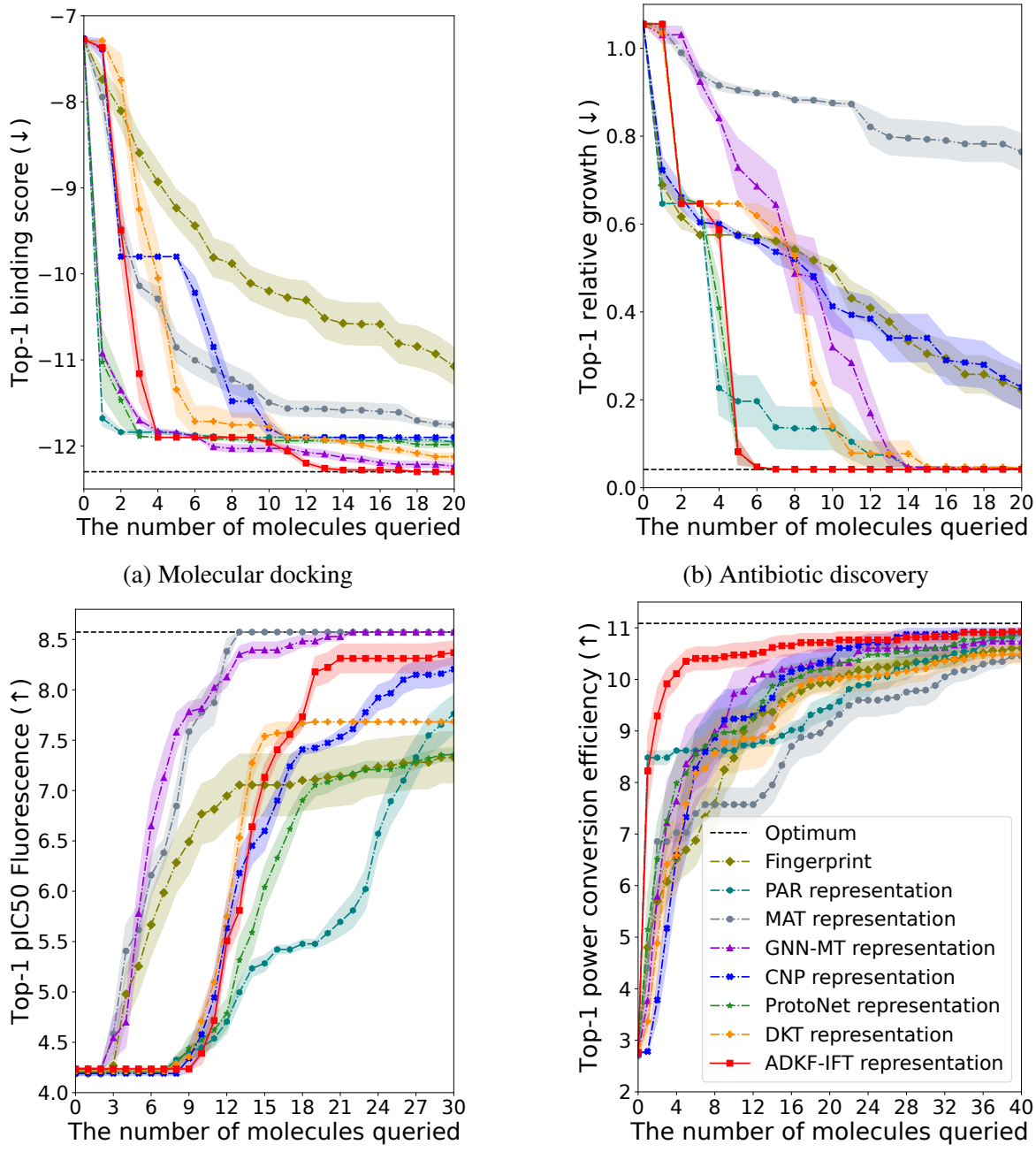
The tasks in FS-Mol can be partitioned into seven sub-benchmarks by Enzyme Commission (EC) number (Webb et al., 1992), which enables sub-benchmark evaluation within the entire benchmark. Ideally, the best method should be able to perform well across all sub-benchmarks. Table 3.5 shows the test performance of top performing methods on all sub-benchmarks at support set size 64 (the median of all considered support sizes) for both the classification and regression tasks. The results indicate that, in addition to achieving best overall performance, ADKF-IFT achieves the best performance on all sub-benchmarks for the regression task and on more than half of the sub-benchmarks for the classification task.

Meta-Testing Costs

We found that ADKF-IFT was ~ 2.5x slower than CNP, ProtoNet, and DKT, but still much faster than GNN-MAML and PAR in terms of wall-clock time on a pre-defined set of FS-Mol classification tasks during the meta-testing phase. However, we would like to point out that meta-testing cost is not an important metric for the experiments considered in this section, as real-time adaptation is not required in drug discovery applications, but could be of interest if ADKF-IFT were to be deployed in other settings. More details of the meta-testing costs of different meta-learning methods can be found in Appendix B.5.

3.5.3 Out-of-Domain Molecular Property Prediction and Optimisation

Finally, we demonstrate that the feature representation learned by ADKF-IFT is useful not only for in-domain molecular property prediction tasks but also for out-of-domain (OOD) molecular property prediction and optimisation tasks. For this, we perform experiments involving finding molecules with best desired target properties within given OOD datasets using Bayesian optimisation (BO) with a GP surrogate model operating on top of compared



(a) Molecular docking

(b) Antibiotic discovery

(c) Antiviral drug design

(d) Material design

Figure 3.6 Mean top-1 target values with standard errors as a function of the number of molecules queried for all compared feature representations on four out-of-domain molecular optimisation tasks.

Table 3.7 Mean predictive performance (test NLL) with standard errors of a GP operating on top of each compared feature representation on the four out-of-domain molecular design tasks.

Feature representation	Out-of-domain molecular design task			
	Molecular docking	Antibiotic discovery	Antiviral drug design	Material design
Fingerprint	$\underline{1.138 \pm 0.014}$	1.669 ± 0.075	$\mathbf{4.601 \pm 0.086}$	1.091 ± 0.011
PAR	1.270 ± 0.019	2.185 ± 0.115	4.840 ± 0.086	1.283 ± 0.017
MAT	1.528 ± 0.028	2.390 ± 0.104	4.797 ± 0.088	2.198 ± 0.063
GNN-MT	1.994 ± 0.050	3.692 ± 0.225	6.399 ± 0.181	7.254 ± 0.217
CNP	1.493 ± 0.028	2.537 ± 0.162	5.005 ± 0.086	1.741 ± 0.043
ProtoNet	1.147 ± 0.013	1.615 ± 0.094	5.060 ± 0.086	1.032 ± 0.009
DKT	1.167 ± 0.012	$\underline{1.602 \pm 0.073}$	4.975 ± 0.092	$\underline{1.026 \pm 0.009}$
ADKF-IFT	$\mathbf{1.137 \pm 0.011}$	$\mathbf{1.496 \pm 0.043}$	$\underline{4.781 \pm 0.087}$	$\mathbf{0.996 \pm 0.007}$

feature representations. We compare them on four representative molecular design tasks (Achdout et al., 2022; García-Ortegón et al., 2022; Hachmann et al., 2011; Stokes et al., 2020) outside of FS-Mol, as summarised in Table 3.6. For the GP surrogate model, we use the Tanimoto kernel for the fingerprint representation (with radius 2 and 2,048 bits based on count simulation) and Matérn-$5/2$ kernel without automatic relevance determination (ARD) but with a log-normal prior over the lengthscale centred at the median heuristic initialisation (Garreau et al., 2017) for all the other compared feature representations. We use the expected improvement (EI) acquisition function (Jones et al., 1998) with a query-batch size 1 for BO. We re-fit the base kernel parameters using all available data points at the beginning of each BO iteration. All compared feature representations are extracted using the models trained on the FS-Mol dataset from scratch in Section 3.5.2, except for the pretrained MAT representation and the fixed fingerprint representation. We repeat each BO experiment 20 times, each time starting from 16 randomly sampled molecules from the worst ~ 700 molecules within the dataset. Figure 3.6 shows that the ADKF-IFT representation enables fastest discovery of top performing molecules for the molecular docking, antibiotic discovery, and material design tasks. For the antiviral drug design task, although the ADKF-IFT representation underperforms the MAT and GNN-MT representations, it still achieves competitive performance compared to other baselines.

Table 3.7 explicitly reports the regression predictive performance of a GP operating on top of each compared feature representation for these four out-of-domain molecular design tasks. The configuration of the GP is the same as that in the BO experiments. We report test negative log likelihood (NLL) averaged over 200 support/query random splits (100 for each of the support set sizes 32 and 64). The results show that the ADKF-IFT representation has the best test NLL on the molecular docking, antibiotic discovery, and material design tasks, and ranks second on the antiviral drug design task.

3.6 Discussion

In this chapter, we proposed Adaptive Deep Kernel Fitting with Implicit Function Theorem (ADKF-IFT), a novel framework for fitting deep kernels that interpolates between meta-learning and conventional deep kernel learning. ADKF-IFT meta-learns a feature extractor across tasks such that the task-specific GP models estimated on top of the extracted feature representations can achieve the lowest possible prediction error on average. ADKF-IFT is implemented by solving a bilevel optimisation objective via implicit differentiation. We showed that ADKF-IFT is a unifying framework containing DKL and DKT as special cases. We demonstrated that ADKF-IFT learns generally useful feature representations, achieving state-of-the-art performance on a variety of real-world few-shot molecular property prediction tasks and on out-of-domain molecular property prediction and optimisation tasks, outperforming a wide range of single-task, multi-task, pre-training, and meta-learning methods. We believe that ADKF-IFT could potentially be an important method to produce well-calibrated models for fully-automated high-throughput experimentation in the future.

Having said that, one potential factor that could limit the use case of ADKF-IFT is that it requires a meta-dataset of many related tasks for meta-training the feature extractor, which is not always available in practice. Furthermore, the inner optimisation procedure is a major bottleneck in ADKF-IFT since it needs to be performed until convergence in every training and inference step. One way to make the training and inference procedure more efficient is to amortise the inner optimisation with a learned task embedding of the support set through a neural network set function. However, we found that learning informative task embeddings was difficult in practice since the aggregation step in set functions tend to cause information loss, which matches the observations in prior work (Garnelo et al., 2018; Tossou et al., 2019; Zaheer et al., 2017).

In the next chapter, we will further investigate the theoretical properties of neural network representations learned across multiple tasks, particularly focusing on recovering canonical feature representations from observed data that align with the ground-truth underlying data generating process.

Chapter 4

Probabilistic Multi-Task Regression for Identifiable Representation Learning

This chapter is based on Chen et al. (2024a):

- Wenlin Chen*, Julien Horwood*, Juyeon Heo, José Miguel Hernández-Lobato. **Leveraging Task Structures for Improved Identifiability in Neural Network Representations.** *Transactions on Machine Learning Research (TMLR)*, 2024.

Both co-first authors (*) contributed equally to method development, code implementation, experimentation and manuscript writing under the supervision of the last author. Moreover, I proved all theorems for this work.

In the previous chapter, we demonstrated that learning a shared feature representation across multiple tasks combined with an adaptive probabilistic regression head for each task could be useful for achieving state-of-the-art empirical results in various low-data downstream tasks, including few-shot molecular property prediction and optimisation tasks. This chapter further investigates the theoretical properties of neural network representations learned across multiple tasks, showing that canonical representations of the data may be recovered in the multi-task learning setting by explicitly modelling the task distribution via a probabilistic approach. In such cases, we show that linear identifiability is achievable in the general multi-task regression setting. Furthermore, we show that the existence of a task distribution, which defines a conditional prior over latent factors, reduces the equivalence class for identifiability to point-wise permutations and scaling of the true latent factors. This is a much stronger result than the linear identifiability result and block-wise permutation identifiability result in prior work. Crucially, when we further assume an underlying structure over these tasks, our approach enables simple maximum marginal likelihood optimisation for recovering canonical representations

from the linearly identifiable representations obtained in the multi-task regression neural network. Empirically, we find that this straightforward probabilistic modelling procedure enables our method to outperform previous identifiable representation learning approaches for both synthetic data and real-world molecular data.

4.1 Motivation and Overview

Multi-task regression is a common problem in machine learning, which naturally arises in many scientific applications such as molecular property prediction (Stanley et al., 2021) and machine learning force fields (Jacobson et al., 2023). Despite this, most deep learning approaches to this problem attempt to model the relationships between tasks through heuristic approaches, such as fitting a shared neural network via end-to-end training, in an attempt to capture the joint structures between tasks. Beyond lacking a principled approach to modelling task relationships, these approaches fail to account for how we may expect the latent factors[1] for related tasks to change. This chapter demonstrates that by leveraging certain assumptions about the relationships between the latent factors of the data *across* tasks, in particular that they vary in their causal and spurious relationships with the target variables, we can recover the canonical latent factors up to permutations and scaling.

A common assumption in the causal representation learning literature, known as the sparse mechanism shift hypothesis (Perry et al., 2022; Schölkopf, 2022; Schölkopf et al., 2021), states that changes across tasks arise from sparse changes in the underlying causal mechanisms. While we do not operate directly on structural causal models, our result arises by similarly considering the implications of sparse changes in the causal graph defining a multi-task learning setting. We accomplish this by first extending the theory of identifiability in supervised learning to the multi-task regression setting for identifiability up to linear transformations (i.e., weak identifiability). We then propose a new probabilistic approach to identifying neural network representations up to permutations and scaling (i.e., point-wise identifiability), by leveraging the assumed causal structures of the underlying latent factors for each task. We empirically validate our model's ability to recover the ground-truth latent structure of the data both in simulated settings where data is generated from our model and for real-world molecular data. This contrasts with current state-of-the-art approaches such as Khemakhem et al. (2020a); Lu et al. (2022), whose assumptions also fit our assumed data generating process but which are difficult to train effectively and only identifiable up to block permutations and scaling of the sufficient statistics of their exponential family priors. The key contributions of this chapter are summarised as follows.

[1]We will use the terms "latent factors/variables" and "(data/feature) representations" interchangeably.

1. In contrast to prior work (Fumero et al., 2023; Lachapelle et al., 2023) which relates meta/multi-task learning to identifiability via explicit sparsity constraints, this work expands these conceptual connections *beyond sparsity constraints* by considering the shared causal structure between tasks. This *significantly* reduces the number of tasks needed to recover the true representations.
2. Our method extends previous identifiability results by relaxing the requirement of energy-based parametrisation for the likelihood (Corollary 4.3.3) and resolving the *point-wise indeterminacies* of the latent factors (Theorem 4.3.5).
3. Our model extends the applicability of conditional prior models to discriminative settings at test time, since our identifiability result does *not* require conditioning on the target variable during inference.
4. To our knowledge, our approach is the first to propose a conditionally *factorised* prior model which can achieve identifiability via optimising the *exact* marginal likelihood. This leads to significantly improved empirical identifiability results in our experiments with synthetic and real-world data despite the probabilistic assumptions of our model.
5. While many works have shown that spurious correlations are a failure case of deep learning and focus on eliminating them (Arjovsky et al., 2019; Eastwood et al., 2022; Kirichenko et al., 2023; Krueger et al., 2021; Lu et al., 2022; Rojas-Carulla et al., 2018), we leverage spurious features to *improve* the robustness of learned representations in the multi-task regression setting through our identifiability results.

4.2 Preliminaries and Related Work

4.2.1 Disentanglement and Independent Component Analysis

The notion of optimising for disentangled representations gained traction in the recent unsupervised deep learning literature when it was proposed that this objective may be sufficient to improve desirable attributes such as interpretability, robustness, and generalisation (Bengio et al., 2013; Chen et al., 2016; Higgins et al., 2017). However, the notion of disentanglement alone is not intrinsically well-defined, as there may be many disentangled representations of the data which are seemingly equally valid. Thus, it is not clear apriori that this criterion is sufficient to achieve the above desiderata (Locatello et al., 2019). In the identifiable representation learning literature, the *correct* disentangled representation is assumed to be the one corresponding to the ground-truth data generating process. Thus, what is required is an *identifiable* representation, which must be equivalent to the causal one for sufficiently expressive model classes. In the linear case, identifiability results exist in the classical literature

for independent component analysis (ICA), which requires non-Gaussian assumptions on the sources for the data (Comon, 1994; Herault and Jutten, 1986).

4.2.2 Conditional Prior Models for Non-Linear ICA

Many extensions of ICA to the non-linear case have been proposed, together with significant theoretical advances. In particular, Hyvarinen et al. (2019) extend this by assuming a conditionally factorised prior over the latent factors given some observed auxiliary variables, and propose a contrastive learning objective for recovering the inverse of the function which generated the observations. Identifiable VAE (iVAE) (Khemakhem et al., 2020a) further extends this to the setting of noisy observations, drawing connections with variational autoencoders (Kingma and Welling, 2013) and enabling direct optimisation via a variational objective. Specifically, iVAE replaces the standard Gaussian prior distribution $p(z)$ over the latent factors (which are invariant to rotations) with a learnable conditional Gaussian distribution $p(z|u)$ whose mean and variance depends on an auxiliary variable u concurrently observed with the data x, such as the label y or time index τ in a time series. Lachapelle et al. (2022) demonstrate that strong identifiability results remain achievable under weaker conditions on the sufficient statistics of the prior *if* the data generating process implies that the latent factors are governed by sparse mechanism shifts. Invariant causal representation learning (iCaRL) (Lu et al., 2022) derives analogous results to iVAE for the case where the prior over the latent factors is a more general non-factorised exponential family distribution, which aligns the model assumption with their assumed causal graph for the underlying data generating process. However, the complex nature of the non-factorised prior in iCaRL requires score matching, which is difficult to optimise in practice. Khemakhem et al. (2020b) explore general conditions for identifiability in energy-based models, and introduce the notion of linear identifiability. This is expanded upon in the context of classification models in Roeder et al. (2021), showing that the representations obtained via the final hidden layer of a neural network may be identifiable up to *linear* transformations when conditioning on the label set. The works of Hälvä et al. (2021); Morioka et al. (2021) both obtain strong identifiability results by exploiting specific temporal or spatial structure in the encoded latent factors and modelling the joint distributions as dynamical systems, however their models do not translate well to the static setting, and their identifiability results remain restricted to non-linear coordinate-wise transformations of the latent variables. While Hyvärinen and Pajunen (1999); Khemakhem et al. (2020a) show that identifiability is not achievable without any form of conditioning in the prior, Kivva et al. (2022); Willetts and Paige (2021) recently extended the results in unsupervised generative models to the case of models with mixture model priors. This can be seen as providing analogous identifiability results to prior methods using conditionally factorised priors, without assumptions on the observability or the dimensionality of the conditioning variable. Nonetheless, these results do

not apply to the exact likelihood, and it remains unclear to what extent the practical consistency and identifiability is achievable when optimising a surrogate variational objective.

4.2.3 Structural Approaches to Identifiability

In contrast, Brady et al. (2023) discuss identifiability results which arise from assumptions on the structure of the mixing function, specifically targeting dual objectives of compositionality with respect to partitions of the latent factors and invertibility of the mixing function. Thus, no distributional assumptions are made on the prior. While this approach has similarities with our proposal by introducing assumptions on how partitions of the latent space evolve with respect to well-defined objects, we propose a general setting which is not restricted to representation learning in visual scenes. Furthermore, by formalising these assumptions within our probabilistic model, we eliminate the need for auxiliary regularisation terms in our optimisation objective.

Recent work (Fumero et al., 2023; Lachapelle et al., 2023) has expanded this area of research to consider the multi-task and meta-learning settings, and thus investigate the connections between identifiability and the structure of the learning problem itself. However, their approach to achieving permutation-identifiable representations relies on introducing heuristic sparsity constraints, such as entropy and L_2-norm regularisers, within a bilevel optimisation objective, which turns out to be difficult to solve both in theory and in practice (Sinha et al., 2017). In addition, their approaches are less applicable in practice since a huge number of tasks are required (more than 10^5 in their experiments). This contrasts with the straightforward, principled and task-efficient optimisation objective arising from our probabilistic model.

4.3 Identifiable Multi-Task Representation Learning

4.3.1 Problem Formulation

The assumptions of the ground-truth data generating process considered in this chapter are encapsulated in the causal graph shown in Figure 4.1, which closely follows the assumptions made in Lu et al. (2022). Note that the input data $x \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$, the target variable[2] $y \in \mathcal{Y} = \mathbb{R}$ and the task index variable $\tau \in \{1, \cdots, N_\tau\}$ are observed variables, and the latent factors $z \in \mathcal{Z} = \mathbb{R}^{d_z}$ $(d_z \leq d_x)$ are unobserved variables. We assume that x is generated by transforming some (unobserved) ground-truth latent factors z^* with some unknown injective mixing function $f_* : \mathcal{Z} \to \mathcal{X}$, i.e., $x = f_*(z^*)$. To incorporate the sparse mechanism shift

[2]Without loss of generality, we assume that the target variable is zero-centred, i.e., $\mathbb{E}(y) = 0$. In practice, this can be achieved by standardising the target variable y.

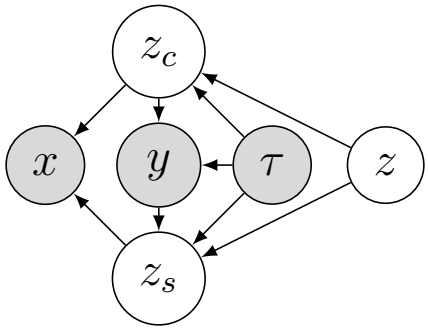


Figure 4.1 Assumed causal graph for the underlying data generating process. For each task τ, we assume that the input data x are generated from the latent factors $z = (z_c, z_s)$. The target variable y is generated by causal latent factors z_c, and z_s are spurious latent factors caused by y. The partition of causal and spurious latent factors can potentially vary across tasks. In molecular property prediction tasks, x corresponds to a molecule, y corresponds to the molecular property to be predicted (e.g., toxicity) in each task τ, and z are the latent factors that control the presence or absence of different substructures in the molecule x.

hypothesis across tasks, we further assume that each task τ has its own partition of the ground-truth latent factors $z^* = (z_c^*, z_s^*)$ into a set of causal latent factors z_c^* and a set of spurious latent factors z_s^*, and such partitions potentially vary across tasks. The target variable y is assumed to be a weighted sum of the causal latent factors z_c, i.e., $y = (w_\tau^*)^\top z^*$, where $w_\tau^* \in \mathbb{R}^{d_z}$ are the ground-truth regression weights for task τ which assign zero weights for the spurious latent factors z_s. Note that there may be latent factors that are uncorrelated with y in some tasks, which can be included within z_c^* but with zero regression weights. The spurious latent factors z_s are assumed to be generated from the target variable y with a different linear correlation function $z_s = y\gamma_\tau$ in each task τ. Our goal is to recover the unobserved ground-truth latent factors z^* given an empirical task distribution $p(\tau)$ over N_τ training tasks and an empirical data distribution $p_d(x, y|\tau)$ for each task indexed by $\tau \in \{1, \cdots, N_\tau\}$.

Overall, our proposed method consists of two stages as illustrated in Figure 4.2. In the first stage (yellow box), we jointly train a multi-task regression network (MTRN) with a feature extractor shared across tasks and N_τ task-specific linear heads using the standard maximum likelihood estimation (MLE) method. We show that upon convergence, the representations learned by the feature extractor are identifiable up to some invertible linear transformation. In the second stage (green box), we use the assumed causal structure across tasks to define a conditional prior over the underlying independent latent factors. We show that this multi-task linear causal model (MTLCM) enables simple maximum marginal likelihood learning for recovering the linear transformation in the representations obtained in the first stage, which reduces the identifiability class to permutations and scaling of the ground-truth latent factors (i.e., point-wise), and automatically disentangles and identifies the causal and spurious latent factors of the target variable from the learned representations.

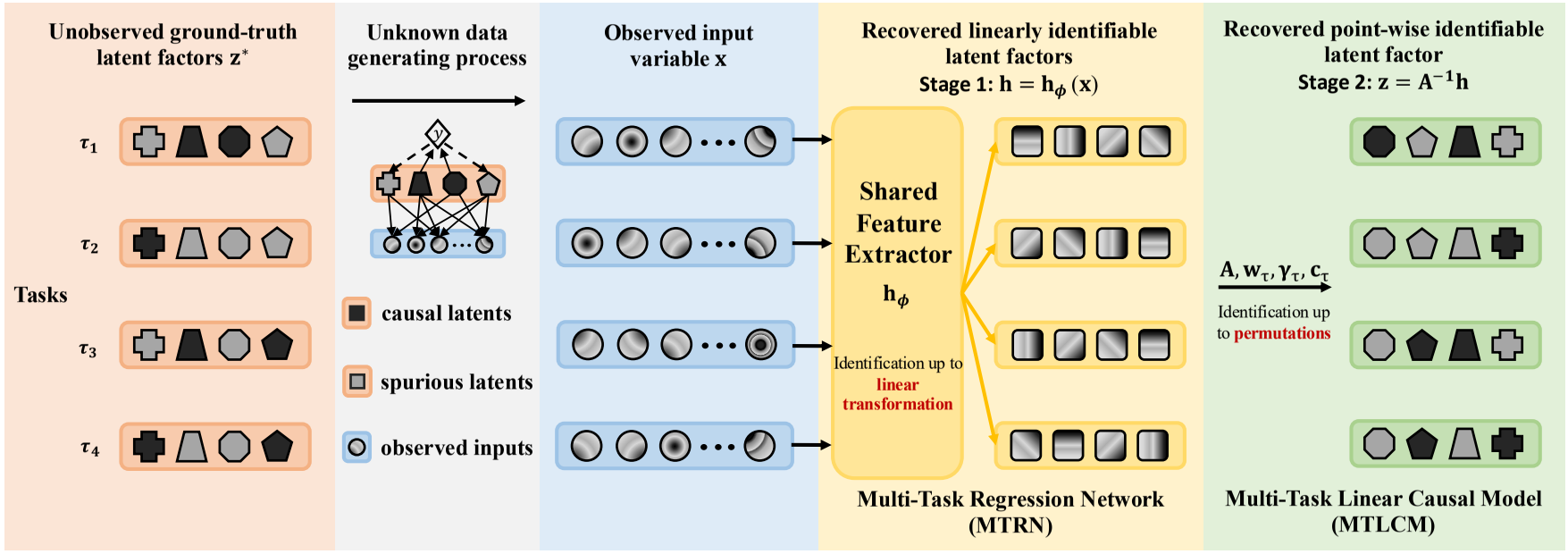


Figure 4.2 The workflow of our proposed method. Shapes are used to track the positions of the ground-truth and recovered latent factors. Colours are used to differentiate between causal and spurious latent factors. We assume that the input data is obtained by transforming the ground-truth latent factors with some mixing function. We show that a multi-task regression network (MTRN) can recover the ground-truth latent factors (i.e., data representations) up to linear transformation and further propose a multi-task linear causal model (MTLCM) to reduce the equivalence class for identifiability to permutations and scaling.

4.3.2 Stage 1: Multi-Task Regression Network

In the first stage, we train a multi-task regression network (MTRN) to recover the ground-truth latent factors up to some invertible linear transformation.

Let $\hat{y}_{\phi,w_\tau}(x) \coloneqq w_\tau^\top h_\phi(x)$ be the output of an MTRN for task τ, where $w_\tau \in \mathbb{R}^{d_z}$ are the regression weights in the linear head for task τ, and $h_\phi(x) \in \mathbb{R}^{d_z}$ is the data representation produced by the feature extractor h_ϕ shared across all tasks with learnable parameters ϕ. As in typical deep non-linear regression settings, the likelihood is assumed to be a conditional Gaussian distribution:

$$p_\theta(y|x,\tau) = \mathcal{N}(y|\hat{y}_{\phi,w_\tau}(x), \sigma^2_{r,\tau}), \tag{4.1}$$

where the mean is parameterised by an MTRN, the variance is fixed to some constant $\sigma^2_{r,\tau}$, and $\theta \coloneqq (\phi, w_1, \cdots, w_{N_\tau})$ denotes all parameters in the MTRN.

We first define linearly identifiable (or weakly identifiable) representations in the multi-task regression setting.

Definition 4.3.1 (Linear identifiability of multi-task learning)**.** *Let $\theta \coloneqq (\phi, \{w_\tau\}_{\tau=1}^{N_\tau})$ and $\theta' \coloneqq (\phi', \{w'_\tau\}_{\tau=1}^{N_\tau})$ be any two sets of multi-task regression model parameters. Then, the data representations are linearly identifiable or weakly identifiable if there exists an invertible matrix $A \in \mathbb{R}^{d_z \times d_z}$ such that*

$$p_\theta(y|x,\tau) = p_{\theta'}(y|x,\tau),\ \forall \tau, x, y \quad \Longrightarrow \quad h_\phi(x) = A h_{\phi'}(x). \tag{4.2}$$

We show that data representations of MTRN are linearly identifiable if we have access to a set of sufficiently diverse tasks measured by the linear dependencies among the regression weights in their linear heads.

Theorem 4.3.2. *Let $\theta \coloneqq (\phi, \{w_\tau\}_{\tau=1}^{N_\tau})$ and $\theta' \coloneqq (\phi', \{w'_\tau\}_{\tau=1}^{N_\tau})$ be any two sets of multi-task regression model parameters such that*

$$p_\theta(y|x,\tau) = p_{\theta'}(y|x,\tau), \quad \forall \tau, x, y. \tag{4.3}$$

Assume that Span(Im(h_ϕ)) $= \mathbb{R}^{d_z}$, i.e., the vectors in the image of the feature extractor h_ϕ span the whole $\mathbb{R}^{d_z}$. Suppose that there exist d_z tasks $\{\tau_i\}_{i=1}^{d_z} \subseteq \{1, \cdots, N_\tau\}$ such that the set of the regression weights $\{w_{\tau_i}\}_{i=1}^{d_z}$ are linearly independent. Then, the data representations of the MTRN are linearly identifiable.

The proof of Theorem 4.3.2 can be found in Appendix C.1.

Following standard practice, we train the MTRN via maximum likelihood estimation (MLE):

$$\theta' \coloneqq \arg\max_{\theta} \ \mathbb{E}_{p(\tau)p_d(x,y|\tau)}[\log p_\theta(y|x,\tau)]. \tag{4.4}$$

Using Theorem 4.3.2, it is straightforward to show that MTRN trained with MLE can recover the ground-truth data representations up to some invertible linear transformation.

Corollary 4.3.3. *Let $h_* : \mathcal{X} \to \mathcal{Z}$ be the ground-truth mapping from input data to the ground-truth latent factors, i.e., $z^* = h_*(x)$. Assume that Span(Im$(h_*)) = \mathbb{R}^{d_z}$. Suppose that there exist d_z tasks $\{\tau_i\}_{i=1}^{d_z} \subseteq \{1, \cdots, N_\tau\}$ such that the set of ground-truth regression weights $\{w^*_{\tau_i}\}_{i=1}^{d_z}$ are linearly independent. Assume that Equation (4.4) has a unique solution and that the optimisation procedure for Equation (4.4) converges to the optimal predictive likelihood*

$$p_*(y|x,\tau) \coloneqq \mathcal{N}(y|(w^*_\tau)^\top h_*(x), \sigma^2_{r,\tau}) \tag{4.5}$$

under standard regularity conditions for MLE estimators (Gurland, 1954), i.e.,

$$p_{\theta'}(y|x,\tau) = p_*(y|x,\tau), \quad \forall \tau, x, y. \tag{4.6}$$

Then, the feature extractor $h_{\phi'}$ is guaranteed to recover the ground-truth latent factors z^* up to some invertible linear transformation A_*, i.e., $h_{\phi'}(x) = A_* h_*(x)$.

Corollary 4.3.3 essentially states that the effective number of tasks defined by the number of independent ground-truth linear heads at least needs to be the same as the number of latent dimensions d_z to guarantee linear identifiability in the multi-task regression setting.

Unlike Roeder et al. (2021) which requires energy-based parametrisation of the output layer, our results are compatible with the more commonly used Gaussian likelihood. Moreover, while Lachapelle et al. (2023)[Proposition 2.2] prove a similar proposition on MLE invariance to linear feature transformations, their proposition is built upon their Assumption 2.1 that the learned feature extractor $h_{\phi'}$ is linearly equivalent to the ground truth feature extractor h_*. However, they do not specify under what conditions this assumption will hold for the MLE objective; they only specify conditions for their bilevel optimisation objective with a sparsity regulariser in their Section 3. In contrast, our Corollary 4.3.3 explicitly reveals such conditions for MLE, i.e., Span(Im$(h_*)) = \mathbb{R}^{d_z}$ and the existence of d_z linearly independent ground-truth task-specific regression weight vectors $\{w^*_{\tau_i}\}_{i=1}^{d_z}$.

4.3.3 Stage 2: Multi-Task Linear Causal Model

In the second stage, we freeze the feature extractor $h_{\phi'}$ learned in the first stage and denote its representations by $h \coloneqq h_{\phi'}(x)$. Corollary 4.3.3 suggests that $h = A_* z^*$ for some invertible

matrix A_*. We propose a *multi-task linear causal model* (MTLCM) to recover the ground-truth latent factors z^* up permutations and scaling from h based on our assumed causal graph in Figure 4.1. The core idea of the MTLCM is to model the change in the causal and spurious latent factors across tasks with learnable task-specific parameters.

Let $\zeta_\tau = \{c_\tau, \gamma_\tau\}$ be a collection of task-specific variables associated with task τ, which are free parameters to be learned from data, where $c_\tau \in \{0,1\}^{d_z}$ are the causal indicator variables which determine the partition of the latent factors $z = (z_c, z_s)$ for the given task τ (i.e., $c_{\tau,i} = 1$ indicates that z_i is a causal latent factor in task τ, and $c_{\tau,i} = 0$ indicates that z_i is a spurious latent factor in task τ), and γ_τ are the linear coefficients used to generate the spurious latent factors z_s from y in task τ.

Conditionally Factorised Structured Prior

Following the standard setting of generative modelling, the prior distribution over causal latent factors z_c are assumed to be a standard Gaussian distribution:

$$p_\zeta(z_c|\tau) = \mathcal{N}(z_c|0, I), \tag{4.7}$$

which depends on the task variable τ since the causal indicator variable c_τ that determines which subset of latent factors are causal varies across tasks.

According to the assumed data generating process, the target variable y is a linear function of the latent data representations z. Following the common setting for the last layer of a non-linear regression neural network, we assume that y is generated from z_c via a linear Gaussian model with the regression weights w_τ masked by the causal indicators c_τ:

$$p_\zeta(y|z_c, w_\tau, \tau) = \mathcal{N}(y|(w_\tau \odot c_\tau)^\top z, \sigma_p^2), \tag{4.8}$$

and that the spurious latent factors z_s are generated from y via another linear Gaussian model with regression weights γ_τ:

$$p_\zeta(z_s|y, \tau) = \mathcal{N}(z_s|y\gamma_\tau, \sigma_s^2 I). \tag{4.9}$$

Since no prior knowledge of the regression weights w_τ is assumed, we marginalize out w_τ from $p_\zeta(y|z_c, w_\tau, \tau)$ over a non-informative prior $p_w(w_\tau)$. This leads a structured conditional prior that factorises over all latent factors z given y and τ:

$$p_\zeta(z|y, \tau) = p_\zeta(z_c|\tau)p_\zeta(z_s|y, \tau) \tag{4.10}$$

$$\coloneqq \mathcal{N}(z|a_\tau, \Lambda_\tau), \tag{4.11}$$

where the mean a_τ and covariance Λ_τ can be compactly expressed as follows:

$$\begin{aligned} a_\tau &:= y\gamma_\tau \odot (1 - c_\tau), \\ \Lambda_\tau &:= \text{diag}(\sigma_s^2(1 - c_\tau) + c_\tau). \end{aligned} \tag{4.12}$$

A detailed derivation of this conditionally factorised prior can be found in Appendix C.3.

Linear Gaussian Likelihood

Since the data representation h learned by MTRN in the first stage is equivalent to z^* up to some linear transformation A_*, we can assume a linear Gaussian likelihood with invertible linear transformation A to recover this linear transformation, similar to the likelihood function in a probabilistic principal component analysis (PCA) model (Tipping and Bishop, 1999):

$$p_A(h|z) = \mathcal{N}(h|Az, \sigma_o^2 I), \tag{4.13}$$

where A is a free parameter to be learned from data, which aims to recover the ground-truth linear transformation A_* for the linearly identifiable representation h obtained before.

Exact Maximum Marginal Likelihood Learning

Let $\psi = (A, \zeta)$ denote all learnable parameters in an MTLCM, which include the linear transformation A and the task-specific parameters $\psi_\tau = \{c_\tau, \gamma_\tau\}$ for all tasks τ. The marginal likelihood for MTLCM is given by

$$\begin{aligned} p_\psi(h|y, \tau) &= \int p_A(h|z) p_\zeta(z|y, \tau) \, \mathrm{d}z \\ &:= \mathcal{N}(h|\mu_\tau, \Sigma_\tau), \end{aligned} \tag{4.14}$$

where the mean μ_τ and covariance Σ_τ have the following closed-form expressions (see Appendix C.4 for a derivation):

$$\begin{aligned} \mu_\tau &:= yA(\gamma_\tau \odot (1 - c_\tau)), \\ \Sigma_\tau &:= A\text{diag}(\sigma_s^2(1 - c_\tau) + c_\tau)A^\top + \sigma_o^2 I. \end{aligned} \tag{4.15}$$

Note that the conditional prior $p(z|y)$ over the latent factors z is typically non-factorised according to the data generating process described in Section 4.3. This is because the causal latent factors z_c are parents of the target variable y, which become correlated when conditioning on y. In order to guarantee block-wise identifiability, iCaRL (Lu et al., 2022) parameterises such non-factorised conditional priors using a ReLU activated energy-based model which is optimised by a hybrid of variational inference and score matching, which turns out to be

difficult to train in practice due to variational over-pruning (Trippe and Turner, 2018) and high computational complexity of score matching (Hyvärinen, 2005). In contrast, our proposed structured conditional prior as shown in Equation (4.10) factorises over all latent factors, which, together with the linear Gaussian likelihood as shown in Equation (4.13), allows us to use exact maximum marginal likelihood optimisation for Equation (4.14) to recover the ground-truth latent factors z^* up to permutations and scaling from the linearly identifiable data representations $h = h_\phi(x)$ learned in the first stage:

$$\psi' \coloneqq \arg\max_{\psi} \; \mathbb{E}_{p(\tau)p_d(x,y|\tau)}[\log p_\psi(h_\phi(x)|y,\tau)]. \tag{4.16}$$

It is worth noting that our method has greater applicability for supervised learning than the methods that rely on a learned probabilistic inverse $q_\omega(z|x,u)$ to extract identifiable latent factors from data such as iVAE (Khemakhem et al., 2020a) and iCaRL (Lu et al., 2022). While these approaches theoretically could apply to learned representations in discriminative settings by letting $u = y$, they are impractical in such contexts since $q_\omega(z|x,y)$ depends on the target variable y which is generally unknown at test time. In contrast, our method does not depend on y at inference time, since the identifiable latent factors can be obtained by applying the inverse linear transformation A learned by MTLCM to the linearly identifiable data representations h produced by the MTRN, i.e., $z = A^{-1}h_\phi(x)$. This enables our model to be applicable to recovering the canonical data representations in the discriminative settings at test time.

Identifiability Theory

We first define point-wise identifiability in the multi-task regression setting.

Definition 4.3.4 (Point-wise identifiability of multi-task representation learning)**.** *Let $\psi = (A, \zeta)$ and $\psi' = (A', \zeta')$ be any two sets of parameters. The latent factors are identifiable up to point-wise permutations and scaling if there exists a permutation and scaling matrix $P \in \mathbb{R}^{d_z \times d_z}$ such that*

$$p_{\psi'}(h|y,\tau) = p_\psi(h|y,\tau), \; \forall h, \tau, y \quad \implies \quad (A')^{-1}h = P(A^{-1}h). \tag{4.17}$$

We show that the latent factors of MTLCM are point-wise identifiable if there are sufficient variations of causal/spurious latent factors across tasks measured by the linear dependencies among the natural parameters of their conditional priors.

Theorem 4.3.5. *Let $u \coloneqq [y, \tau]$ denote the conditioning variable and $k \coloneqq 2d_z$. Assume that the learned and ground-truth linear transformations A and A_* are invertible. Suppose that*

there exist $k+1$ *points* $u_0, u_1, \cdots, u_k$ *such that the matrix*

$$L := [\eta(u_1) - \eta(u_0), \cdots, \eta(u_k) - \eta(u_0)] \tag{4.18}$$

is invertible, where

$$\eta(u) := \begin{bmatrix} \Lambda_\tau^{-1} a_\tau \\ -\frac{1}{2} diag(\Lambda_\tau^{-1}) \end{bmatrix} \in \mathbb{R}^k \tag{4.19}$$

are the natural parameters of our structured conditional prior $p_\zeta(z|u)$*. Assume that Equation* (4.16) *has a unique solution and that the optimisation procedure for Equation* (4.16) *converges to the optimal marginal likelihood*

$$p_*(h|y,\tau) := \mathcal{N}(h|\mu_\tau^*, \Sigma_\tau^*) \tag{4.20}$$

under standard regularity conditions for maximum marginal likelihood estimators (Gurland, 1954), i.e.,

$$p_{\psi'}(h|y,\tau) = p_*(h|y,\tau), \quad \forall h, y, \tau, \tag{4.21}$$

where μ_τ^* *and* Σ_τ^* *are as defined in Equation* (4.15) *but with the ground-truth linear transformation* A_**, ground-truth causal indicators* c_τ^* *and ground-truth spurious coefficients* γ_τ^**. Then, the latent factors recovered by MTLCM are guaranteed to be point-wise identifiable.*

The proof of Theorem 4.3.5 can be found in Appendix C.2. The first part of the proof adapts the proof technique from Khemakhem et al. (2020a) to show identifiability up to block-wise permutations and scaling of the sufficient statistics $[z_i, z_i^2]$. The second part of the proof is novel, which leverages the properties of the linear likelihood as shown in Equation (4.13) to further reduce the block-identifiable equivalence class to point-wise permutations and scaling of the actual ground-truth latent factors. This resolves the point-wise indeterminacies of Khemakhem et al. (2020a); Lu et al. (2022) where their models are only identifiable up to block-wise transformations.

4.4 Empirical Evaluation

This section presents empirical results to validate the ability of **MTLCM** to recover canonical representations up to permutations and scaling for both synthetic and real-world data. We contrast our model with the more general identifiable models of **iVAE** (Khemakhem et al., 2020a) and **iCaRL** (Lu et al., 2022). For a fair comparison, we also consider the multi-task extensions of iVAE and iCaRL, namely **MT-iVAE** and **MT-iCaRL**, which include the task variable τ in the conditioning variables u in their conditional priors $p_\zeta(z|u)$ with the task-specific parameter $\zeta_\tau = \{v_\tau\}$ to be learned from data, which is the counterpart to $\zeta_\tau = \{c_\tau, \gamma_\tau\}$

in our MTLCM but has no explicit interpretations with respect to a causal graph. We note that while the works of Fumero et al. (2023); Lachapelle et al. (2023) also consider methods for identifiability arising from learning across tasks, their approaches effectively implement a meta-learning setting (i.e., require that the support and query sets be disjoint in the bilevel optimisation process). The assumption on task support variability for the latter also requires a much larger number of tasks (more than 10^5 tasks as in their experiments) than what we consider here (around 10^2 tasks). These methods are thus not particularly well suited to comparison with our approach and we omit them from our baselines. Detailed model configurations can be found in Appendix C.5. Each experiment is run until convergence and repeated across 5 random seeds to guarantee reproducibility.

4.4.1 Synthetic Data

We first validate our approach in the situation when the data generating process agrees with the assumptions of our models. For each task, we first sample the causal indicator variables c_τ^*. The causal latent factors z_c^* are then sampled from a standard Gaussian prior. These are then linearly combined according to random weights w_τ^* to produce the target variable y with a task-dependent noise corruption. Finally, the spurious variables z_s^* are generated via different linear weightings γ_τ^* of the target variable y. This mirrors the assumed data generating process described in Section 4.3. For the linear case, we generate observed data using random linear transformations of the ground-truth latent factors. For the non-linear case, we extend this to non-linear transformations parameterised by randomly initialised neural networks and demonstrate that our approach can be combined with the multi-task identifiability result up to linear transformations to recover point-wise permutations and scaling of the ground-truth latent factors.

Linear Synthetic Data

We study the ability of our proposed multi-task linear causal model (MTLCM) to recover the ground-truth latent factors up to point-wise permutations and scaling via the mean correlation coefficient (MCC) as in Khemakhem et al. (2020a). The synthetic data is generated by sampling 200 tasks of 100 data points each. Each task varies in its causal indicator variables c_τ^*, causal weights w_τ^*, and spurious coefficients γ_τ^*. We then transform the ground-truth latent factors z^* with a random invertible matrix A_* shared across all tasks to obtain linearly identifiable representations h. Identifiability in this setting is assessed by directly computing the MCC score between the representations obtained from our MTLCM and the ground-truth latent factors, which is referred to as strong MCC. Since the data is known to be linearly identifiable, we use linear likelihoods for the baselines as well.

Table 4.1 Identifiability performance for recovering the linearly transformed synthetic latent factors measured by strong MCC (%).

#Causal	2					4			
#Latent/Observed	3/3	5/5	10/10	20/20	50/50	5/5	10/10	20/20	50/50
iVAE	87.75±5.02	78.02±0.73	81.36±0.57	82.30±0.27	81.96±0.07	81.67±2.97	74.29±0.30	77.57±0.15	79.79±0.10
iCaRL	75.22±6.40	74.55±2.09	72.37±2.22	79.43±0.52	80.00±1.00	66.98±1.32	66.00±3.00	71.54±1.69	78.67±0.61
MT-iVAE	91.78±8.12	90.14±5.01	**99.89±0.04**	97.90±1.51	90.56±3.18	76.09±7.69	76.36±2.32	98.42±0.88	94.53±2.49
MT-iCaRL	81.09±3.37	71.12±2.97	76.13±0.53	79.26±1.00	81.30±0.84	61.55±1.26	64.04±1.08	72.79±1.92	79.54±0.59
MTLCM	**99.95±0.01**	**99.96±0.01**	99.77±0.16	**99.70±0.16**	**98.97±0.55**	**99.95±0.01**	**99.71±0.21**	**99.51±0.36**	**99.14±0.27**

Table 4.2 Identifiability performance for recovering the non-linearly transformed synthetic latent factors measured by strong MCC (%). The weak MCC (%) for MTRN is also reported as a reference.

#Causal	4			8			12		
#Latent/Observed	20/50	20/100	20/200	20/50	20/100	20/200	20/50	20/100	20/200
iVAE	73.11±1.13	77.42±0.20	76.95±0.31	65.18±1.49	68.66±0.14	69.05±0.17	58.70±0.33	60.33±0.27	59.85±0.31
iCaRL	56.70±3.49	63.29±4.26	58.64±2.83	57.09±2.41	60.66±2.74	61.02±2.43	52.93±2.13	58.80±1.81	54.40±2.54
MT-iVAE	71.78±1.45	80.14±0.37	73.89±2.98	65.44±1.60	69.31±0.35	68.56±0.34	55.79±1.61	60.56±0.23	59.61±0.30
MT-iCaRL	67.57±1.97	70.26±3.22	65.52±0.65	63.37±0.84	63.75±2.19	61.61±1.52	57.13±1.07	60.56±0.15	58.10±1.04
MTLCM	**93.31±1.10**	**97.94±0.71**	**97.44±0.68**	**95.67±0.16**	**98.12±0.75**	**89.05±0.97**	**95.75±0.14**	**96.28±1.20**	**84.28±1.27**
MTRN (weak)	89.38±0.71	96.15±0.91	96.19±0.87	93.96±0.22	97.63±0.79	87.75±0.99	95.14±0.17	96.12±1.27	83.70±1.22

Table 4.1 shows that MTLCM manages to recover the ground-truth latent factors from h up to point-wise permutations and scaling, and the result is scalable as the number of latent factors and the number of causal factors increase. In contrast, iVAE, iCaRL and their multi-task extensions underperform our model by a large margin in most cases. We find that for all tasks, the learned causal indicator variables by MTLCM also exactly match the ground-truth c_τ^*, confirming that our approach automatically disentangles the causal and spurious latent factors. Ablation study for the effects of the learnable parameters and the type of linear transformation can be found in Appendix C.6.

Non-Linear Synthetic Data

A more general analysis of the identifiability of our proposed approach is to consider the extension of the linear experiments to the setting of *arbitrary* transformations of the latent factors. For this, we consider the case where random (non-linear) MLPs are used to transform the ground-truth latent factors z^* into higher dimensional observations x. By Corollary 4.3.3, it is possible to recover linearly identifiable representations h of the data x by training standard multi-task regression networks (MTRNs). Identifiability in this setting is assessed by first performing a canonical correlation analysis (CCA) as in Roeder et al. (2021), which linearly maps the obtained representations such that they maximise the covariance with the ground-truth latent factors. The resulting mapped representations can thus be compared with the ground-truth latent factors via the MCC score. This is referred to as weak MCC, which quantifies the linear identifiability of the learned representations from MTRNs. We further train our MTLCM on the linearly identifiable representations h obtained from the MTRN to obtain identifiable representations up to point-wise permutations and scaling. Identifiability in this setting is assessed by directly computing the MCC score between the representations obtained from our MTLCM and the ground-truth latent factors as in the linear case (i.e., strong MCC). We assess this for various dimensionalities of the observed data and for different settings of the causal variables, where we generate 500 tasks of 200 samples each to improve convergence of the multitask model. The MTRN and the likelihoods in the baselines are parameterised by one-hidden-layer MLPs whose hidden dimensionality are twice the observation dimensionality.

Table 4.2 shows that the strong MCC for MTLCM is able to match the weak MCC for MTRN, confirming the point-wise determinacies of our approach. In contrast, the strong MCC for iVAE, iCaRL and their multi-task extensions significantly underperform MTLCM. Again, we find that for all tasks, the learned causal indicator variables exactly match the ground-truth causal indicator variables c_τ^*.

Table 4.3 Identifiability performance for the latent factors learned on the superconductivity dataset measured by strong MCC (%). The weak MCC (%) for MTRN is also reported as a reference. "−" indicates divergence of optimisation during training.

Method	Latent dim				
	5	10	20	40	80
iVAE	32.87±1.16	33.21±1.04	30.68±0.39	37.41±0.84	45.52±0.81
iCaRL	–	32.23±0.61	35.62±0.40	32.58±2.16	32.19±2.45
MT-iVAE	35.58±1.48	33.54±0.80	31.68±0.32	35.14±0.82	44.49±0.96
MT-iCaRL	–	–	–	–	42.26±2.33
MTLCM	**98.90±0.03**	**96.93±0.12**	**84.56±1.11**	**46.31±0.34**	**48.94±2.16**
MTRN (weak)	98.85±0.03	97.17±0.04	93.23±0.08	78.58±0.09	52.02±0.19

4.4.2 Real-World Molecular Data

We further evaluate our model on two real-world molecular datasets. We assume that the data x is generated by transforming some unknown ground-truth latent factors z^* with some unknown non-linear mixing function f_*. Since z^* are unknown to us, identifiability in this setting is assessed by first training a model five times with different random seeds, then computing the MCC score between the data representations recovered by each pair of these five models, following Khemakhem et al. (2020b). As in Section 4.4.1, we employ the weak MCC score to assess the linear identifiability of the representations h learned by the MTRN and the strong MCC score to assess the point-wise identifiability of the latent factors z recovered by each method. Given that the true latent dimension is unknown, we assess the identifiability of each model at gradually increasing latent dimensions. In practical scenarios, the latent dimension would be selected based on a similar model selection exercise.

Superconductivity Dataset

The superconductivity dataset (Hamidieh, 2018) consists of $21,263$ superconductors. We consider the tasks of regressing 80 readily computed target features (i.e., 80 tasks) such as mean atomic mass, thermal conductivity and valence of the superconductors from their chemical formulae, represented as discrete counts of the atoms present in the molecule. The MTRN and the likelihoods in the baselines are parameterised by two-hidden-layer MLPs with a hidden dimension of 64.

Table 4.3 shows that the strong MCC for our MTLCM is greater than 0.96 and is able to match the weak MCC for the MTRN when the dimensions of the latent representations are 5 and 10, showing that our method manages to recover canonical latent representations for the superconductors. Interestingly, the strong MCC score for the MTLCM rapidly decreases as

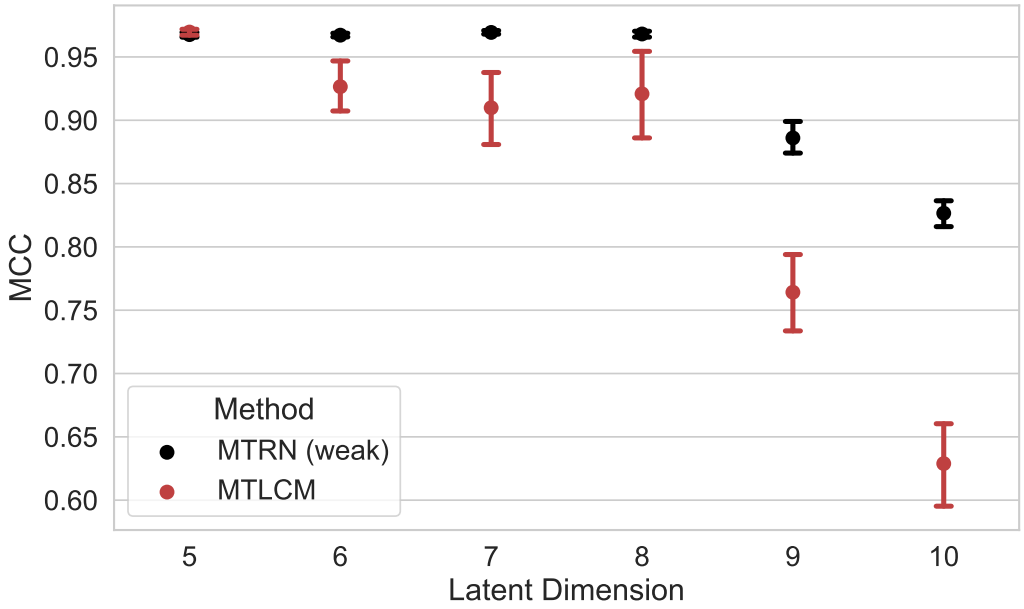


Figure 4.3 Identifiability performance for the latent factors learned on the QM9 dataset.

we increase the number of latent factors in the model, suggesting that there are at most 10-20 effective tasks out of the 80 tasks used for this data. In sharp contrast, all baseline models fail to recover identifiable latent factors for the superconductors in all cases as their strong MCC scores do not exceed 0.4. There are several settings where optimisation diverged during training, since VAE and score-based models are generally difficult to train on discrete inputs of chemical formulae.

QM9 Dataset

The QM9 dataset (Ramakrishnan et al., 2014; Ruddigkeit et al., 2012) is a popular benchmark for molecular prediction tasks consisting of $134,000$ enumerated organic molecules of up to nine heavy atoms together with a set of 12 calculated quantum chemical properties (i.e., 12 tasks). In contrast to the more artificial superconductivity dataset, the QM9 dataset enables us to assess the feasibility of achieving identifiable representations in the context of highly non-trivial quantum chemical properties which are highly relevant to their pharmacological profile. Accurately modelling this dataset requires us to capture potential three-dimensional atomic interactions, allowing us to assess the translation of our theoretical results to more complex equivariant graph neural network architectures. For this reason, we use an equivariant graph neural network (EGNN) (Satorras et al., 2021) as the feature extractor for the MTRN. This enables the model to incorporate positional features of each atom while respecting physical symmetries such as equivarance to rotation and translation. Given that the graph autoencoders

proposed in Satorras et al. (2021) and prior works (Kipf and Welling, 2016; Liu et al., 2019a; Simonovsky and Komodakis, 2018) do not provide a means of jointly decoding the feature and adjacency matrices, we do not consider the iVAE and iCaRL baselines for this dataset.

Figure 4.3 shows that the linear identifiability achieved from the MTRN implies that identifiability is achievable up to eight latent features, after which there is a sharp decline in MCC. The implication of this observation is that there exist some redundancies between tasks (i.e., the number of effective tasks is less than the total number of tasks), which limit the maximal identifiable latent dimension. This is clearly the case for certain tasks. For example, prediction of the HOMO-LUMO gap can be directly obtained as a result of the difference between HOMO (highest occupied molecular orbital energy) and LUMO (lowest unoccupied molecular orbital energy) values. Nonetheless, the MTLCM is able to closely approximate the weak MCC score up to eight latent factors, always surpassing a score of 0.9, demonstrating its ability to recover point-wise identifiable representations in the context of real-world molecular datasets.

4.5 Discussion

In this chapter, we proposed a novel perspective on the problem of multi-task identifiable representation learning by exploring the implications of explicitly modelling task structures via probabilistic multi-task regression models. We showed that this resulted in new identifiability theory for linear equivalence classes in the general case of deep multi-task regression. Furthermore, while spurious correlations have been shown to be a failure case of deep learning in many recent works, we demonstrated that such latent spurious signals might in fact be leveraged to *improve* the ability of a model to recover more robust disentangled representations (i.e., point-wise identifiability). In particular, when the latent space is explicitly represented as consisting of a partitioning of causal and spurious features per task, the linear identifiability result of the multi-task regression may be further reduced to identifiability up to point-wise permutations and scaling under sufficient variability conditions of the tasks. Empirically, we confirmed that the theoretical results held for both linear and non-linear synthetic data and for two real-world molecular datasets of superconductors and organic small molecules. We anticipate that this may reveal new research directions for the study of both representation learning and synergies with probabilistic and multi-task learning methods, and hope that these methods will produce robust feature representations that can generalise to unseen tasks.

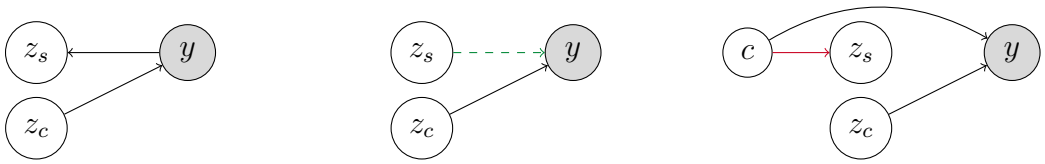
Below, we discuss some of the main assumptions underlying our approaches, as well as their implications and limitations.

1. Our linear identifiability result requires the dimensionality of the learned feature representations be smaller than the number of effective (or "independent") training tasks as measured by the linear dependencies among the weights of their regression heads. This can limit the expressivity of the learned representations if we do not have access to a sufficiently large number of independent training tasks. Furthermore, if the dimensionality of the representations is too small, the model family may not contain the ground-truth data generating process, and therefore it would be impossible to recover the ground-truth model that has generated the observed data.

2. Our point-wise identifiability result requires that there is at least one spurious latent factor in some tasks and that the causal/spurious split cannot be identical across tasks. Otherwise, the prior $p_\zeta(z|u)$ over the latent factors may not have sufficient auxiliary information to guarantee point-wise identifiability. These requirements can be a limiting factor for real-world applications, since they may not always be satisfied in practice.

3. While our model requires certain linear and Gaussian assumptions, we would like to point out that they are well-justified. In Stage 1, the Gaussian likelihood in Equation (4.1) is a standard choice for the predictive distribution in the final layer of a (non-linear) deep regression network. In Stage 2, the Gaussian prior over the causal latent factors in Equation (4.7) follows the standard setting of generative modelling. The linear Gaussian regression models for the causal latent factors in Equation (4.8) and for the spurious latent factors in Equation (4.9) are analogous to the standard predictive distribution for the final layer of regression neural networks. The linear Gaussian likelihood in Equation (4.13) follows the standard probabilistic liner PCA model (Tipping and Bishop, 1999), which is a natural choice given the linear identifiability result arising from the MTRN in Stage 1.

4. Our model proposes that the correlations between latent factors and the regression target for each task be modelled as a partitioning of causal and spurious influences. We acknowledge that indeed our assumed direct edge $y \to z_s$ in Figure 4.1 does not in general capture all possible non-causal correlations between latent factors and the target variable, since the Reichenbach principle (Reichenbach and Morrison, 1956) states that non-causal correlations can originate from either a common cause (i.e., confounders) or an anti-causal/spurious relationship (as assumed in this work). However, we argue that there are many situations where the proposed model can be useful in practice, even if it does not explicitly model the confounders in full generality, since this anti-causal assumption employed in our paper has been used in prior work (Lu et al., 2022) and is well-documented in real-world examples. For example, in most drug discovery campaigns, molecules to be tested are selected based on some structural similarities



(a) Anti-causal (b) Uncorrelated (c) Confounding

Figure 4.4 Illustration of the possible relationships between a latent factor z_s and target variable y for a given task. Cases (a) and (b) are captured by our model. Note that (b) can be handled by our model by treating z_s as a causal latent variable with zero regression weight on the dashed green arrow $z_i \to y$. Case (c) is not captured by our model, because although the unobserved confounder variable c can be viewed as a latent factor, the red arrow $c \to z_s$ cannot be captured by our model.

to an originally promising molecule (based on the quantity to be estimated, e.g., drug potency). Structural molecule features are then likely to be spuriously correlated with the regression target due to their selection criteria, without actually being involved in the drug's mechanism of action. Having said that, one could in principle consider other cases: one where there is no correlation between a latent variable and the target variable, or one where the correlation between a latent variable and the target arises from a confounding variable. These possibilities are depicted graphically in Figure 4.4. We note that the former case could be handled by our model by treating it as a causal variable with a regression weight of zero. In the latter case, this confounding variable would then itself be a latent variable with a causal association to the target variable and a latent factor, which indeed cannot be captured by our model.

5. While the causal and probabilistic assumptions of our approach do not constitute the most general conceivable case, we note that there is an inherent trade-off between full generality and tractability. Indeed, prior work which may theoretically allow for more general causal or probabilistic models typically entail approximations in the optimisation. Further, the empirical results on real-world data of Section 4.4 suggest that our approach may indeed be robust to moderate model mis-specification.

So far, we have leveraged probabilistic inference to improve the data-efficiency and identifiability of molecular representation learning. In the next chapter, we will explore a complementary perspective: using deep learning to enhance probabilistic inference. We will show that deep learning has the potential to improve the scalability and efficiency of sampling-based probabilistic inference for multi-modal probability distributions.

Chapter 5

Diffusion-Inspired Training of Deep Generative Models for Enhanced Sampling

This chapter is mainly based on He et al. (2025):

- Jiajun He*, Wenlin Chen*, Mingtian Zhang*, David Barber, José Miguel Hernández-Lobato. **Training Neural Samplers with Reverse Diffusive KL Divergence**. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2025.

In additional, this chapter also contains materials from Chen et al. (2024b):

- Wenlin Chen*, Mingtian Zhang*, Brooks Paige, José Miguel Hernández-Lobato, David Barber. **Diffusive Gibbs Sampling**. *International Conference on Machine Learning (ICML)*, 2024.

For each work, co-first authors (*) contributed equally to method development, code implementation, experimentation and manuscript writing under the supervision of the remaining authors.

In the previous two chapters, we showed that probabilistic inference approaches can be leveraged to improve the data efficiency and identifiability of deep neural networks for improved representation learning. This chapter explores the opposition direction in the reciprocal relationship between deep learning and probabilistic inference, demonstrating that deep learning can in turn enhance the efficiency of probabilistic inference. Specifically, we leverage ideas from generative deep learning to the improve sampling-based probability inference for *unnormalised* probability distributions in the case where ground-truth samples are *unavailable*. Inspired by diffusion models, we propose to minimise the reverse Kullback-Leibler (KL) divergence along the marginal diffusion trajectories of the densities of a deep

generative model and the target distribution. The resulting objective is referred to as reverse *diffusive KL* (DiKL) divergence, which enables deep generative models to explore all high density regions of the target distribution. This is in contrast to the well-known *mode-collapse* behaviour found in the traditional reverse KL divergence. We derive a tractable gradient estimator for training deep implicit models with this objective, enabling efficient generation of independent samples from target distributions in *one step*. We find that our method enhances sampling efficiency across various Boltzmann distributions, including both synthetic multi-modal densities and many-body particle systems.

5.1 Motivation and Overview

Sampling from unnormalised probability distributions is an essential and challenging research problem with wide applications in machine learning, statistics, and natural sciences. We consider a target distribution with an analytical but *unnormalised* density function:

$$p_d(x) = \frac{\exp(-E(x))}{Z}, \tag{5.1}$$

where $x \in \mathcal{X}$ is the random variable of interest and $E : \mathcal{X} \to \mathbb{R}$ is a lower-bounded differentiable energy function. The normalising constant involves an integral:

$$Z = \int \exp(-E(x))\,\mathrm{d}x, \tag{5.2}$$

which is typically intractable for complex target distributions in practice. The gradient of the log target density is known as the *score function*, which is independent of Z:

$$\nabla_x \log p_d(x) = -\nabla_x E(x), \tag{5.3}$$

Note that the score function can be evaluated at any input x, since E is assumed to be differentiable. This assumption is commonly satisfied in various practical applications, such as Bayesian inference (Welling and Teh, 2011), energy-based generative modelling (Song and Ermon, 2019), and molecular dynamics simulation (Noé et al., 2019).

The goal of the sampling problem is to draw independent samples $x \sim p_d(x)$ from the target distribution efficiently, which can be used to approximate the expectations of functions of interest over the target distribution $p_d(x)$ as defined in Equation (2.35). For example, such expectations may correspond to physical quantities that describe the macroscopic behaviours of molecular dynamics. A common approach to sampling from $p_d(x)$ involves designing Markov Chain Monte Carlo (MCMC) samplers (Neal et al., 2011). Apart from the standard MCMC algorithms presented in Section 2.1.2, advanced approaches that build on top of

existing MCMC samplers, such as parallel tempering (PT) (Geyer and Thompson, 1995; Surjanovic et al., 2022; Swendsen and Wang, 1986; Syed et al., 2022) and diffusion Gibbs sampler (DiGS) (Chen et al., 2024b), have been proposed to accelerate the mixing speed of MCMC samplers. However, for high-dimensional multi-modal target distributions, these methods still take a long time to converge and need to simulate a very long chain for the generated samples to be uncorrelated (Chen et al., 2024b; Pompe et al., 2020). This presents significant challenges in large-scale simulation problems.

Alternatively, one can leverage generative deep learning to amortise the sampling process. This approach emulates the sampling process by approximating the target distribution $p_d(x)$ with a deep generative model $p_\theta(x)$, which can directly produce independent samples from $p_d(x)$. Such models are often referred to as *neural samplers* (Arbel et al., 2021; di Langosco et al., 2022; Levy et al., 2018; Wu et al., 2020). Training a neural sampler involves learning the model parameters θ, which is usually achieved by minimising a divergence between $p_\theta(x)$ and $p_d(x)$. In the sampling setting, we only assume access to the unnormalised probability density function $p_d(x) \propto \tilde{p}_d(x) = \exp(-E(x))$ defined by an energy function $E(x)$ without any ground-truth samples, which is different from the usual generative modelling setting where ground-truth samples $\mathcal{D} = \{x_n\}_{n=1}^N \sim p_d(x)$ are available for model training.

A common objective for training neural samplers is the reverse KL (R-KL) divergence due to its tractability. However, the most significant limitation of R-KL is the *mode-collapse* phenomenon due to its mode-seeking behaviour (Bishop, 2006): when the target distribution $p_d(x)$ contains multiple distant modes, the model $p_\theta(x)$ trained by R-KL will underestimate the variance of $p_d(x)$ and can only capture few modes. This is undesirable since many important target distributions, such as the Bayesian posteriors (Welling and Teh, 2011) and Boltzmann distributions (Noé et al., 2019), are characterised by a large number of distant modes.

This chapter presents a novel approach to training neural samplers with *diffusive KL* (DiKL) divergence. DiKL convolves both the target and the model densities with Gaussian convolution kernels, allowing for better connectivity and merging of distant modes in the noisy space. Notably, DiKL is still a valid divergence between the model density $p_\theta(x)$ and the *original* target density $p_d(x)$, which allows us to learn the original target distribution with better mode-covering capability. The key contributions of this chapter are summarised as follows.

1. We introduce a novel paradigm for training neural samplers with the reverse DiKL divergence (Section 5.3.1), which encourages neural samplers to approximate the target distribution more effectively by promoting mode coverage compared to the traditional reverse KL divergence (Section 5.3.2).

2. We derive a tractable gradient estimator for the reverse DiKL divergence, enabling efficient training of neural samplers with the proposed objective in practice (Section 5.3.3).

Additionally, we show how equivariance can be achieved with this estimator when dealing with physical systems with symmetry constraints (Section 5.5.2).

3. We evaluate our approach on both synthetic multi-modal targets and many-body particle systems, finding that it outperforms or matches the performance of previous state-of-the-art models with reduced training and sampling costs (Section 5.5).

5.2 Preliminary: Kullback-Leibler Divergence

5.2.1 Definition of KL Divergence

Kullback-Leibler (KL) divergence is a type of statistical divergence that measures the statistical distance between two probability distributions. For two distributions with density functions p and q, KL divergence is defined as

$$\mathrm{KL}(p||q) = \int (\log p(x) - \log q(x)) p(x) \, \mathrm{d}x. \tag{5.4}$$

Note that KL divergence is well-defined if $q(x) = 0$ implies $p(x) = 0$ for all $x \in \mathcal{X}$. It can be shown that KL divergence has the following two important properties (Bishop, 2006).

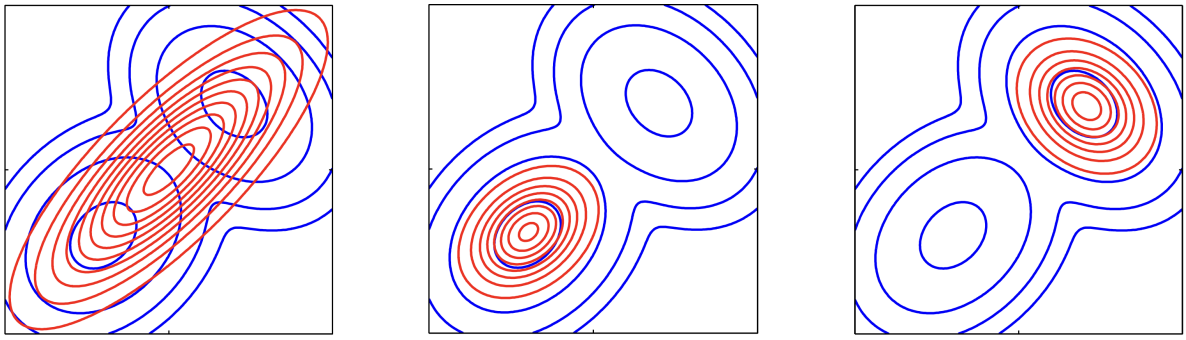
1. It is always non-negative:
$$\mathrm{KL}(p||q) \geq 0, \quad \forall p, q. \tag{5.5}$$

2. It equals zero if and only if the two density functions are identity almost everywhere (a.e.):
$$\mathrm{KL}(p||q) = 0 \quad \Longleftrightarrow \quad p = q \text{ (a.e.)}. \tag{5.6}$$

However, it is worth noting that KL divergence is not a valid distance metric, since it is asymmetric (i.e., $\mathrm{KL}(p||q) \neq \mathrm{KL}(q||p)$) and does not satisfy the triangle inequality in general. Nevertheless, $\mathrm{KL}(p||q)$ measures the difference between two distributions in terms of the expected excess surprise from using the distribution q as a model when the true distribution is p. Due to its computational tractability and nice probabilistic interpretation, KL divergence is widely used in machine learning as loss functions for model training, such as fitting generative models to a given target distribution.

5.2.2 Forward KL Minimisation

Given a target distribution $p_d(x)$, it is a common practice to fit a generative model $p_\theta(x)$ to the target distribution $p_d(x)$ by minimising the *forward* KL (F-KL) divergence when ground-truth



(a) Forward KL (b) Reverse KL (c) Reverse KL

Figure 5.1 A comparison between model training with forward and reverse KL divergences. Red contour lines depict a uni-Gaussian model, and blue contour lines depict a mixture of Gaussians target distribution. The figures show that F-KL exhibits the mass-covering property, while reverse KL exhibits the mode-collapse phenomenon. The figures are reproduced from Bishop (2006).

samples from $p_d(x)$ are available:

$$\mathrm{KL}(p_d||p_\theta) = \int (\log p_d(x) - \log p_\theta(x)) p_d(x)\,\mathrm{d}x \tag{5.7}$$

$$= -\int p_d(x) \log p_\theta(x)\,\mathrm{d}x + \text{const.}, \tag{5.8}$$

which is equivalent to maximising the log marginal density $p_\theta(x)$ as discussed in Section 2.3.2, since the entropy of the target distribution $p_d(x)$ is a constant independent of the model parameters θ. Forward KL (F-KL) divergence has a nice *mass-covering* property (Bishop, 2006), as shown in Figure 5.1a, which encourages the model to explore all high density areas of the target distribution.

This objective is particularly suitable for models with analytically tractable marginal densities $p_\theta(x)$, such as normalising flows. For other latent variable models with intractable marginal densities like VAEs, it is possible to derive a tractable upper bound of F-KL:

$$\mathrm{KL}(p_d||p_\theta) \leq \mathrm{KL}(q_\phi(z|x)p_d(x)||p_\theta(x|z)p_z(z)), \tag{5.9}$$

which is equivalent to the variational lower bound of the log marginal density $\log p_\theta(x)$ as shown in Equation (2.78). As discussed in Section 2.3.2, variational approaches often suffer from its own problems despite tractability.

5.2.3 Reverse KL Minimisation

For the sampling problem considered in this chapter, we only assume access to the unnormalised density of the target distribution $p_d(x) \propto \exp(-E(x))$. Since no ground-truth samples

are available, F-KL is intractable in this setting. A common alternative choice is to minimise the *reverse* KL (R-KL) divergence:

$$\begin{aligned}\mathrm{KL}(p_\theta||p_d) &= \int (\log p_\theta(x) - \log p_d(x)) p_\theta(x)\,\mathrm{d}x \\ &= \int (\log p_\theta(x) + E(x)) p_\theta(x)\,\mathrm{d}x + \log Z,\end{aligned} \tag{5.10}$$

where $\log Z$ is a constant independent of x or θ. The integration over $p_\theta(x)$ can be approximated by the Monte Carlo method using samples from the model $p_\theta(x)$ with the reparametrisation trick. In contrast to F-KL, R-KL exhibits the *mode-seeking* behaviour as shown in Figure 5.1b and Figure 5.1c, which typically results in the trained model $p_\theta(x)$ collapsing to a small number of modes in a multi-modal target distribution (Bishop, 2006).

Again, this objective can be directly optimised for models with analytically tractable marginal densities. For more general latent variable model with intractable marginal densities, Zhang et al. (2019) derive a tractable variational upper bound of the R-KL:

$$\mathrm{KL}(p_\theta||p_d) \leq \mathrm{KL}(p_\theta(x|z)p_z(z)||q_\phi(z|x)p_d(x)), \tag{5.11}$$

where $q_\phi(z|x)$ is a learnable variational distribution. This variational upper bound contrasts with the more commonly studied F-KL upper bound as shown in Equation (5.9) and Equation (2.78). While this approach circumvents the intractability of $\log p_\theta(x)$, it again suffers from the common challenges and limitations of variational inference.

Alternatively, Li and Turner (2018); Luo et al. (2023); Shi et al. (2018); Song et al. (2020) derive an analytical expression of the gradient of R-KL with respect to model parameters θ:

$$\nabla_\theta \,\mathrm{KL}(p_\theta||p_d) = \int p_\theta(x)(\nabla_x \log p_\theta(x) - \nabla_x \log p_d(x)) \frac{\partial x}{\partial \theta}\,\mathrm{d}x, \tag{5.12}$$

where the score function $\nabla_x \log p_\theta(x)$ of the model can be approximated by training a score network using the model samples with score matching (Hyvärinen, 2005). This enables fitting latent variable models to unnormalised densities by R-KL minimisation without any variational approximation.

5.3 Diffusive Kullback-Leibler Divergence

Diffusion models are high-quality deep generative models trained using ground-truth samples from a target distribution (Ho et al., 2020; Sohl-Dickstein et al., 2015; Song et al., 2021b). A critical success factor for diffusion models is the Gaussian convolution trick that gradually transforms a complicated multi-modal target distribution $p_d(x)$ into a simple uni-modal

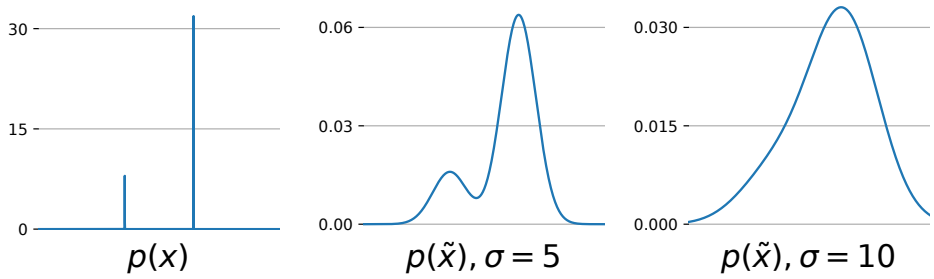


Figure 5.2 We convolve a Gaussian kernel $\mathcal{N}(\tilde{x}|x, \sigma^2)$ with $\sigma \in \{5, 10\}$ to the original distribution $p(x)$. This demonstrates that Gaussian convolution can bridge modes and even reduce the number of modes as the variance of the Gaussian increases.

Gaussian distribution to encourage exploration of the support of $p_d(x)$ (Chen et al., 2024b; Huang et al., 2024; Lee et al., 2021): it turns out that Gaussian convolution can potentially reduce the number of modes, since it effectively convexifies any functions to its convex envelope (Mobahi and Fisher, 2015). Figure 5.2 provides a visualisation of a toy example, showing that by increasing the variance of the Gaussian convolution kernel, we can bridge modes or even reduce the number of modes in the original multi-modal distribution.

5.3.1 Definition of DiKL Divergence

To construct a valid divergence that leverages Gaussian convolutions, one can convolve two distributions $p(x)$ and $q(x)$ with the same Gaussian kernel:

$$k(\tilde{x}|x) = \mathcal{N}(\tilde{x}|\alpha x, \sigma^2 I), \tag{5.13}$$

and then define the KL divergence between the convolved distributions:

$$\tilde{p}(\tilde{x}) = \int k(\tilde{x}|x)p(x)\,\mathrm{d}x, \tag{5.14}$$

$$\tilde{q}(\tilde{x}) = \int k(\tilde{x}|x)q(x)\,\mathrm{d}x. \tag{5.15}$$

This type of divergence construction method is known as the spread divergence (Zhang et al., 2020).

Definition 5.3.1. *The spread KL (SKL) divergence between two distributions with density functions p and q is defined as the KL divergence between the convolved distributions $\tilde{p}$ and $\tilde{q}$:*

$$SKL(p||q) := KL(\tilde{p}||\tilde{q}) = KL(p * k||q * k), \tag{5.16}$$

where $*$ denotes the convolution operator defined as

$$\tilde{\pi} \coloneqq \pi * k \coloneqq \int k(\tilde{x}|x)\pi(x)\,\mathrm{d}x \tag{5.17}$$

for any density function π.

SKL is a theoretically well-defined statistical divergence, i.e., $\mathrm{SKL}_k(p||q) \geq 0$ and $\mathrm{SKL}_k(p||q) = 0$ if and only if $p = q$ (a.e.) for any Gaussian kernel k (Zhang et al., 2020).

In practice, the choice of k is crucial for model training with spread KL divergence, and selecting the optimal kernel is a challenging problem. Intuitively, for a given fixed contraction factor $\alpha \in [0, 1]$, a large noise standard deviation σ will reduce the barriers of adjacent modes in the target distribution $p_d(x)$. Similarly, with a fixed σ, reducing α will bring the modes closer. Different choices of the Gaussian kernel k can result in different properties in the corresponding SKL, which will be further discussed in 5.3.2. Inspired by the recent success of diffusion-based deep generative models, instead of selecting one k, one can use a sequence of Gaussian kernels with different contracting factors and noise levels to construct a *multi-level* spread KL divergence, which we refer to as *diffusive KL* (DiKL) divergence.

Definition 5.3.2. *The diffusive KL (DiKL) divergence between two distributions with density functions p and q is defined as a weighted sum of a sequence of convolve distributions obtained by different Gaussian convolution kernels:*

$$\mathit{DiKL}_{\mathcal{K}}(p||q) \coloneqq \sum_{t=1}^{T} w(t)\mathit{KL}(p * k_t||q * k_t) \leq \mathit{KL}(p||q), \tag{5.18}$$

where $w(t)$ is a positive scalar weighting function that sum to one, and $\mathcal{K} = \{k_1, \cdots, k_T\}$ is a set of Gaussian convolution kernels denoted as $k_t(x_t|x) = \mathcal{N}(x_t|\alpha_t x, \sigma_t^2 I)$.

The fact that DiKL divergence is a weighted sum of multiple spread KL divergence with different Gaussian kernels implies that it is a valid statistical divergence, i.e., $\mathrm{DiKL}_{\mathcal{K}}(p||q) \geq 0$ and $\mathrm{DiKL}_{\mathcal{K}}(p||q) = 0$ if and only if $p = q$ (a.e.). Below is a sketch of proof for a single kernel k_t following Zhang et al. (2020); the extension to multiple kernels is straightforward.

$$\begin{aligned}
\mathrm{DiKL}_{k_t}(p||q) = 0 \quad &\iff \quad p * k_t = q * k_t \text{ (a.e.)} \\
&\iff \quad \mathcal{F}[p * k_t] = \mathcal{F}[q * k_t] \\
&\iff \quad \mathcal{F}[p]\mathcal{F}[k_t] = \mathcal{F}[q]\mathcal{F}[k_t] \\
&\iff \quad \mathcal{F}[p] = \mathcal{F}[q] \\
&\iff \quad p = q \text{ (a.e.)}.
\end{aligned} \tag{5.19}$$

Moreover, DiKL is a lower bound of the KL divergence; see Appendix D.1 for a proof.

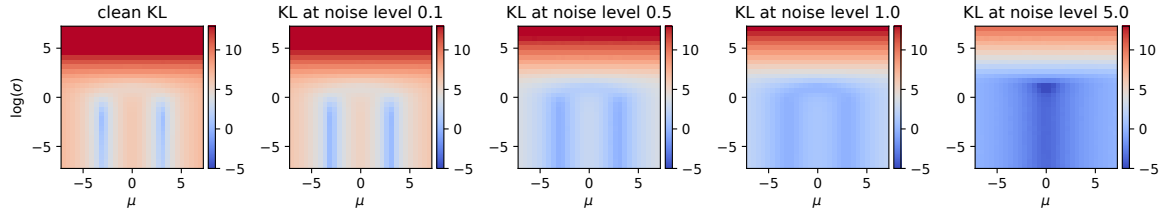


Figure 5.3 Heatmap of (log scale) R-KL and R-DiKL at different noise levels between a Gaussian model (with mean parameter μ and standard deviation parameter σ) and a two-mode mixture of Gaussians target distribution in 1D against different values of the model parameters μ and σ. At lower noise levels (or in the extreme case, the standard R-KL), the divergence is highly mode-seeking, with the model favouring either one of the two modes in the target distribution. In contrast, the KL divergence becomes more mode-covering at higher noise levels, encouraging the model to cover both modes in the target distribution.

5.3.2 Reverse DiKL Encourages Mode-Covering

Unlike R-KL which has a mode-seeking nature, reverse DiKL (R-DiKL) promotes better mode coverage. This section provides an intuitive explanation to illustrate how this is achieved. Suppose that we have a 1D mixture of Gaussians (MoG) target with two components, whose density is given by

$$p_d(x) = \frac{1}{2}\mathcal{N}(x|-3, 0.1^2) + \frac{1}{2}\mathcal{N}(x|3, 0.1^2). \tag{5.20}$$

For simplicity, we fit the target $p_d(x)$ with a 1D Gaussian model $p_\theta(x) = \mathcal{N}(x|\mu, \sigma^2)$, which only contains two parameters $\theta = \{\mu, \sigma\}$. This allows us to visualise the values of log R-KL and log R-DiKL at different noise levels against these two parameters to develop a better understanding of the reverse DiKL objective. The results are shown in Figure 5.3. It can be seen that at lower noise levels (or in the extreme case, R-KL), the divergence is highly mode-seeking, with the model favouring either one of the two modes in the target distribution similar to the behaviour of R-KL. However, perhaps surprisingly, at higher noise levels, DiKL becomes more mass-covering, forcing the mean parameter μ to converge towards the mean of the two modes and the variance parameter σ^2 to cover both modes. This behaviour explains why DiKL encourages the model to cover more modes: higher noise levels push the model to explore adjacent modes, while lower noise levels prevent the model from forgetting previously discovered modes and refine the models around these modes.

5.3.3 Training Neural Samplers with Reverse DiKL

This section presents a practical gradient estimator for training neural samplers with R-DiKL. We focus on training neural samplers defined by a deep implicit model due to its flexibility (as

discussed in Section 2.3.2):

$$p_\theta(x) = \int \delta(x - f_\theta(z)) p_z(z)\, \mathrm{d}z, \tag{5.21}$$

where $p_z(z) = \mathcal{N}(z|0, I)$ and $f_\theta : \mathcal{Z} \to \mathcal{X}$ is a generator parameterised by a standard non-invertible neural network. Unlike the conventional KL divergence which requires the model $p_\theta(x)$ to have a valid density function (Arjovsky et al., 2017) as discussed in Section 2.3.2, SKL and DiKL are well-defined even for singular distributions (e.g., Dirac delta), as discussed in Zhang et al. (2020). Therefore, we can still train these deep implicit models using DiKL even though the model density $p_\theta(x)$ may not be well-defined.

We now explore how to train such a deep generative model $p_\theta(x)$ to fit the unnormalised target density $p_d(x)$ using R-DiKL, denoted as $\mathrm{DiKL}_{\mathcal{K}}(p_\theta||p_d)$. For simplicity, we first consider R-DiKL with a single kernel k_t; the extension to a linear combination of multiple kernels is straightforward. The R-DiKL between the model density $p_\theta(x)$ and the target density $p_d(x)$ is defined as[1]:

$$\begin{aligned} \mathrm{DiKL}_{k_t}(p_\theta||p_d) &\coloneqq \mathrm{KL}(p_\theta * k_t||p_d * k_t) \\ &= \int p_\theta(x_t) \left(\log p_\theta(x_t) - \log p_d(x_t)\right) \mathrm{d}x_t, \end{aligned} \tag{5.22}$$

where $p_\theta(x_t) \coloneqq \int k(x_t|x) p_\theta(x)\, \mathrm{d}x$ and $p_d(x_t) \coloneqq \int k(x_t|x) p_d(x)\, \mathrm{d}x$. The integration over $p_\theta(x_t)$ can be approximated using the Monte Carlo method by first sampling $x' \sim p_\theta(x)$ from the neural sampler model and then adding noise to obtain $x_t \sim k(x_t|x')$.

Inspired by Luo et al. (2024); Poole et al. (2022); Wang et al. (2023b), we can derive the analytical gradient of R-DiKL in Equation (5.22) with respect to the model parameters θ:

$$\begin{aligned} \nabla_\theta \mathrm{DiKL}_{k_t}(p_\theta||p_d) &= \nabla_\theta\, \mathrm{KL}(p_\theta * k_t||p_d * k_t) \\ &= \int p_\theta(x_t) \left(\nabla_{x_t} \log p_\theta(x_t) - \nabla_{x_t} \log p_d(x_t)\right) \frac{\partial x_t}{\partial \theta}\, \mathrm{d}x_t, \end{aligned} \tag{5.23}$$

The derivation of Equation (5.23) can be found in Appendix D.2. The Jacobian term $\frac{\partial x_t}{\partial \theta}$ can be efficiently computed by the vector-Jacobian product (VJP) with automatic differentiation. However, both score functions in Equation (5.23), $\nabla_{x_t} \log p_\theta(x_t)$ and $\nabla_{x_t} \log p_d(x_t)$, are intractable to compute directly.

To address this, we approximate these scores using denoising score matching (DSM) (Vincent, 2011) and mixed score identity (MSI) (De Bortoli et al., 2024; Phillips et al., 2024), respectively.

[1]To avoid notation overloading, we slightly abuse p_d to represent the density function for both the original clean target density and the convolved noisy target density $p_d * k_t$. We distinguish them by their arguments: $p_d(x)$ denotes the original target density, while $p_d(x_t)$ refers to the convolved density. This also applies to the kernel k_t and model density p_θ.

Specifically, we estimate the noisy model score $\nabla_{x_t} \log p_\theta(x_t)$ by training a score network with DSM using samples from the model, and estimate the noisy target score $\nabla_{x_t} \log p_d(x_t)$ by MSI with Monte Carlo estimation. Below, we explain these two estimators in detail.

Estimating the Noisy Model Score $\nabla_{x_t} \log p_\theta(x_t)$ with Denoising Score Matching

As discussed in Section 2.3.2, denoising score matching (DSM) (Vincent, 2011) has been successfully used in training score-based diffusion models (Song et al., 2021b). In our case, we can train a time-conditioned score network[2] $s_\phi(x_t)$ to approximate $\nabla_{x_t} \log p_\theta(x_t)$ by minimising the DSM loss with respect to the surrogate score model parameters ϕ:

$$\mathcal{L}_{\text{DSM}}(\phi) = \mathbb{E}_{\mathcal{U}(t)p_\theta(x)k(x_t|x)}[\lambda(t)\|s_\phi(x_t) - \nabla_{x_t} \log k(x_t|x)\|^2], \tag{5.24}$$

The expectation can be approximated using the Monte Carlo method by sampling from the neural sampler model $x' \sim p_\theta(x)$ followed by sampling $x'_t \sim k(x_t|x')$. Note that the model parameters θ are fixed in this stage. Once trained, we can plug $s_\phi(x_t)$ into Equation (5.23) as part of the gradient estimator for R-DiKL.

Estimating the Noisy Target Score $\nabla_{x_t} \log p_d(x_t)$ with Mixed Score Identity

To estimate the gradient defined in Equation (5.23), we also need to estimate the noisy target score $\nabla_{x_t} \log p_d(x_t)$. Since no samples from the target distribution $p_d(x)$ are available, we can no longer use DSM to estimate this score function. Fortunately, we have access to the unnormalised target density and its score function $\nabla_x \log p_d(x) = -\nabla_x E(x)$, which allows us to estimate this score by target score identity (TSI) (De Bortoli et al., 2024).

Proposition 5.3.3 (**Target Score Identity**). *For any translation-invariant convolution kernel $k(x_t|x) = k(x_t - \alpha_t x)$, we have*

$$\nabla_{x_t} \log p_d(x_t) = \frac{1}{\alpha_t} \int \nabla_x \log p_d(x) p_d(x|x_t) \, \mathrm{d}x, \tag{5.25}$$

where $p_d(x|x_t) \propto k(x_t|x)p_d(x)$ is the denoising posterior distribution.

The proof of Proposition 5.3.3 can be found in Appendix D.3.1. In practice, the TSI estimator has larger variance when the Gaussian kernel $k(x_t|x)$ has larger variance, while the DSI estimator from Proposition 2.3.1 exhibits higher variance when $k(x_t|t)$ has smaller variance. To address this, De Bortoli et al. (2024); Phillips et al. (2024) propose a convex combination of the DSI and TSI to interpolate between them, favouring TSI when $k(x_t|x)$ has smaller variance and DSI when $k(x_t|x)$ has larger variance, thus minimising the overall variance of the estimator. We refer to this estimator as the mixed score identity (MSI).

[2]For simplicity of notation, we drop the time argument t in $s_\phi(x_t, t)$ and simply write $s_\phi(x_t)$ instead.

Proposition 5.3.4 (Mixed Score Identity). *Using a Gaussian convolution kernel* $k(x_t|x) = \mathcal{N}(x_t|\alpha_t x, \sigma_t^2 I)$ *with a variance-preserving (VP) scheme* $\sigma_t^2 = 1 - \alpha_t^2$*, and a convex combination of TSI and DSI with coefficients* α_t^2 *and* $1 - \alpha_t^2$*, respectively, we have*

$$\nabla_{x_t} \log p_d(x_t) = \int (\alpha_t(x + \nabla_x \log p_d(x)) - x_t) p_d(x|x_t) \, \mathrm{d}x. \tag{5.26}$$

The proof of Proposition 5.3.4 can be found in Appendix D.3.2. This identity estimates the noisy target score $\nabla_{x_t} \log p(x_t)$ based on the original clean target score $\nabla_x \log p_d(x)$ which can be directly evaluated. The resulting estimate can be plugged into Equation (5.23) as the other part of the gradient estimator for R-DiKL.

However, in order to use this estimator, we also need to obtain samples from the denoising posterior $p_d(x|x_t)$ to approximate the integration over x in Equation (5.26). We notice that the denoising posterior distribution $p_d(x|x_t)$ is proportional to the joint distribution over the clean and noisy data: $p_d(x, x_t) = k(x_t|x)p_d(x)$, taking the form

$$p_d(x|x_t) \propto \exp\left(-E(x) - \frac{\|\alpha_t x - x_t\|^2}{2\sigma_t^2}\right), \tag{5.27}$$

which has a tractable score (Chen et al., 2024b; Gao et al., 2021; Huang et al., 2024):

$$\nabla_x \log p_d(x|x_t) = -\nabla_x E(x) - \frac{\alpha_t(\alpha_t x - x_t)}{\sigma_t^2}. \tag{5.28}$$

Therefore, common score-based sampler, such as HMC, MALA and AIS as introduced in Section 2.1.2, can be directly employed to draw samples from the denoising posterior $p(x|x_t)$. Notably, compared to sampling from the original target distribution $p_d(x) \propto \exp(-E(x))$, incorporating the additional quadratic term as shown in Equation (5.27) improves the Log-Sobolev conditions, which significantly enhances the convergence speed of samplers based on Langevin dynamics (Huang et al., 2024; Vempala and Wibisono, 2019).

It is worth noting that it is not crucial to have a perfect denoising posterior sampler. This is because our method essentially works in a bootstrapping manner: the posterior samples improve the model, and a better model in turn brings the posterior samples closer to the true target. As a result, although we did not extensively tune the hyperparameters of our posterior sampler in our experiments, we found that our method still worked well with only a few MCMC steps in practice. Having said that, accurate posterior sampling may further improve the convergence rate of the model.

Algorithm 2 Training Neural Samplers with Reverse Diffusive KL (R-DiKL) Divergence

Input: target distribution $p_d(x) \propto \exp(-E(x))$, Gaussian kernel hyperparameters $\{(\alpha_t, \sigma_t)\}_{t=1}^T$; the number N_ϕ of training steps for the score network $s_\phi(x_t)$; weighting functions $\lambda(t)$ and $w(t)$; randomly initialised model parameters θ, ϕ.
repeat
  # Inner loop: train the score network $s_\phi(x_t)$ by DSM
  for $i \in [1, \cdots, N_\phi]$ **do**
    $z \sim p_z(z)$, $x \leftarrow f_\theta(z)$
    $t \sim \mathcal{U}\{1, \cdots, T\}$, $\varepsilon \sim \mathcal{N}(0, I)$
    $x_t \leftarrow \alpha_t x + \sigma_t \varepsilon$
    Update ϕ with $\nabla_\phi \lambda(t) \| s_\phi(x_t) - \nabla_{x_t} \log k(x_t|x) \|^2$ ▷ DSM training
  end for
  # Outer loop: train the neural sampler $f_\theta(z)$ by R-DiKL
  $z \sim p_z(z)$, $x \leftarrow f_\theta(z)$
  $t \sim \mathcal{U}\{1, \cdots, T\}$, $\varepsilon \sim \mathcal{N}(0, I)$
  $x_t \leftarrow \alpha_t x + \sigma_t \varepsilon$
  $x'^{(1:L)} \sim p_d(x|x_t)$ ▷ posterior sampling
  $d_p \leftarrow \frac{1}{L} \sum_{l=1}^L (\alpha_t (x'^{(l)} + \nabla \log p_d(x'^{(l)})) - x_t)$ ▷ MSI estimator
  $\ell \leftarrow w(t)\, \texttt{stopgrad}(s_\phi(x_t) - d_p)^\top x_t$ ▷ surrogate loss for VJP
  Update θ with $\nabla_\theta \ell$
until convergence
Output: neural sampler parameters θ

Training Procedure

We summarise the whole procedure of training neural samplers with R-DiKL in Algorithm 2[3]. In short, our training algorithm forms a nested loop.

- Inner loop: we train a surrogate time-conditioned score network $s_\phi(x_t)$ to estimate the noisy model score $\nabla_{x_t} \log p_\theta(x_t)$ by minimising the DSM loss.
- Outer loop: we first estimate the noisy target score $\nabla_{x_t} \log p_d(x_t)$ with MSI and posterior sampling, and then update the parameters θ of the neural sampler using the gradient as in Equation (5.23) with the estimated noisy target score and noisy model score.

This results in an Expectation-Maximisation (EM) style training algorithm. It might be tempting to think that this nested training procedure imposes a high computational burden. Fortunately, we found that the inner loop typically converged within 50-100 steps in practice, minimally affecting the overall training cost.

[3]For clarity, Algorithm 2 presents the training procedure with a batch size of 1.

5.4 Related Work

5.4.1 Neural Samplers

General One-Step Neural Samplers

Various approaches for training (non-invertible) one-step generators as neural samplers with R-KL have been explored in the literature, which employ different approximate inference techniques for gradient estimation, including score matching (SM) (Li and Turner, 2018; Luo et al., 2023; Song et al., 2020), variational inference (VI) (Shi et al., 2018; Yin and Zhou, 2018; Zhang et al., 2019) and Stein's method (Hu et al., 2018). These methods typically struggle for multi-modal target distributions due to the mode-seeking property of R-KL.

Flow-Based Neural Samplers

The first neural sampler which focuses on sampling from Boltzmann distributions is a flow-based sampler called Boltzmann generator (Noé et al., 2019), which trains normalising flows with R-KL and exact gradient estimation due to its tractable model density. It tends to miss modes when the target distribution is multi-modal due to the limited expressivity of normalising flows and the mode-seeking behaviour of R-KL.

Flow annealed importance sampling bootstrap (FAB) (Midgley et al., 2023) is the state-of-the-art flow-based neural sampler, which is trained by minimising the alpha-2 divergence $D_{\alpha=2}(p_d||p_\theta) \propto \int p_d(x)^2/p_\theta(x)\,\mathrm{d}x$ that exhibits mass-covering property (Minka, 2005) and corresponds to the variance of the IS weight $w_{\text{IS}}(x) = p_d(x)/p_\theta(x)$. FAB estimates this intractable objective using AIS with the flow model density $p_\theta(x)$ as the initial distribution and $r(x) \propto p_d(x)^2/p_\theta(x)$ as the target distribution, which minimises the variance of the IS estimator for the alpha-2 divergence. Note that FAB employs a prioritised replay buffer to memorise the regions that have been explored.

Diffusion-Based Neural Samplers

An increasing number of works have been exploring diffusion-based samplers. The Gibbs-style sampler (Chen et al., 2024b; Grenioux et al., 2024; Zhang et al., 2023a) constructs an iterative forward-backward sampling procedure between the original clean data space and the noisy diffusion space to bridge distant modes in the target distribution. The path integral sampler (PIS) (Zhang and Chen, 2022) and the denoising diffusion sampler (DDS) (Vargas et al., 2023) align the forward and backward paths by optimising the KL divergence over the entire path measure. GFlowNet-based samplers (Bengio et al., 2023; Zhang et al., 2024) extend these approaches to objectives with local information, such as sub-trajectory balance

and detailed balance. Controlled Monte Carlo diffusions (CMCD) (Nusken et al., 2024) learns an escorted transport between interpolants from the prior distribution to the target distribution by matching the KL or log-variance divergence between the forward and backward path measures. Non-equilibrium transport samplers (NETS) (Albergo and Vanden-Eijnden, 2024) learn a similar escorted transport with physics-informed neural networks (Sun et al., 2024) or action matching loss (Neklyudov et al., 2023). Further improvements include combining these samplers with sequential Monte Carlo (Chen et al., 2025a), or incorporating MCMC to improve buffer samplers (Sendera et al., 2024). However, all these methods typically involve simulating an SDE with numerical integration during training, which is not scalable to high-dimensional target distributions.

Iterated denoising energy matching (iDEM) (Akhound-Sadegh et al., 2024) trains a score network to approximate the noisy target score $\nabla_{x_t} \log p_d(x_t)$ estimated by TSI, which is one of the most scalable diffusion-based neural samplers due to its simulation-free nature. It also uses a reply buffer to balance exploration and exploitation.

5.4.2 Variational Score Distillation

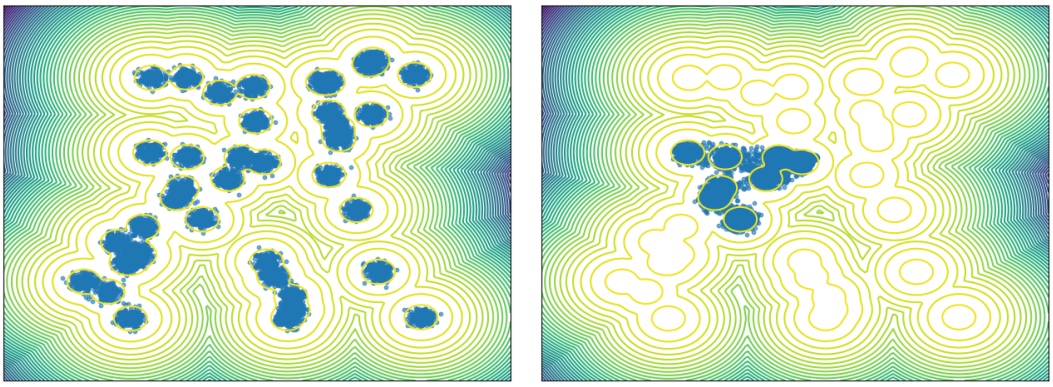
Variational score distillation (VSD) is a promising approach to distil knowledge from pre-trained diffusion models. VSD shares a similar idea to DiKL in the sense that it distils a pre-trained diffusion model into a one-step generator by adding noise to the densities of both models before computing the KL divergence, which has been successfully applied to training 3D generative models (Poole et al., 2022; Wang et al., 2023b) and distilling diffusion models (Luo et al., 2024; Xie et al., 2024). However, in VSD, the score functions of p_d and $p_d * k_t$ are provided by a pre-trained diffusion model, which is different from our setting where those score functions are estimated by MSI, as we only assume access to the unnormalised target density without any ground-truth samples for pre-training such diffusion models.

5.5 Empirical Evaluation

This section presents empirical evaluation results for the performance of neural samplers trained by R-DiKL as described in Algorithm 2 on a synthetic multi-modal target distribution (Section 5.5.1) and three many-body particle systems (Section 5.5.2). Detailed experimental setup for each experiment can be found in Appendix D.5.

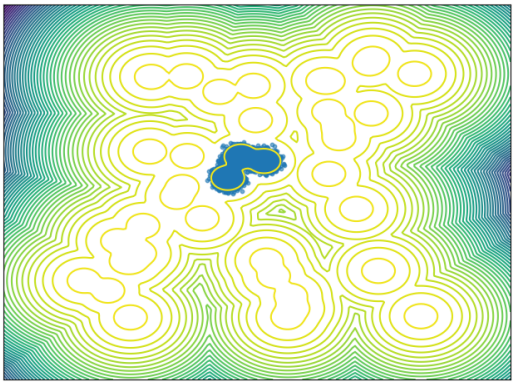
5.5.1 Synthetic Multi-Modal Target Distribution

We compare **R-DiKL** with different types of state-of-the-art neural samplers on a mixture of 40 Gaussians (MoG-40) target distribution with 40 modes in 2D following Midgley et al.

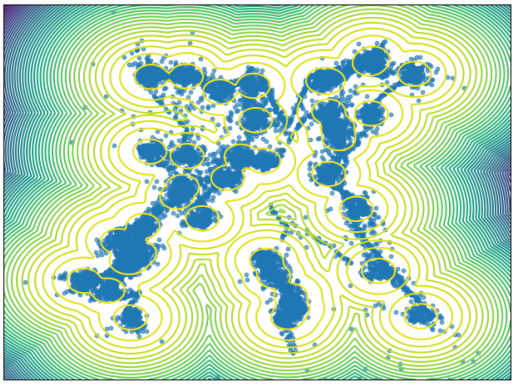


(a) Ground Truth

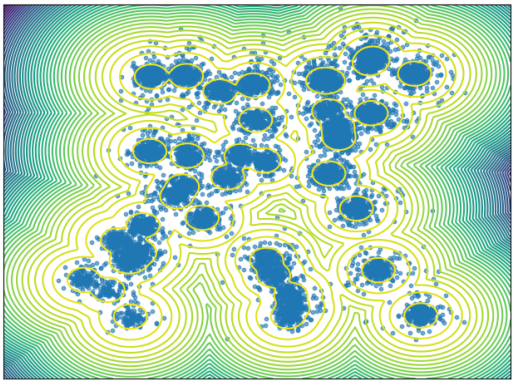
(b) R-KL-SM



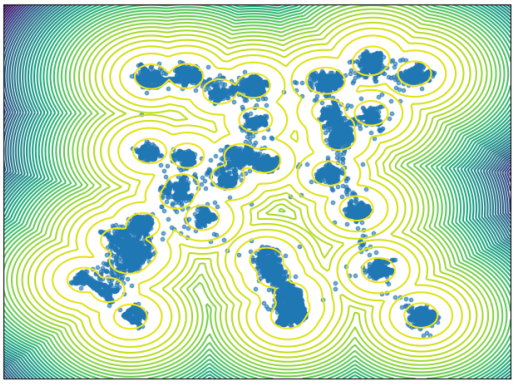
(c) R-KL-VI



(d) FAB



(e) iDEM



(f) R-DiKL

Figure 5.4 Visualisation of samples generated by all compared neural samplers on MoG-40. We train each method for 2.5 hours, which allows all to converge. FAB and iDEM use replay buffers as in Akhound-Sadegh et al. (2024); Midgley et al. (2023). The high-density regions of this target are within $[-50, 50]$. All methods were trained on the original scale, except for iDEM, which is normalised to $[-1, 1]$ following Akhound-Sadegh et al. (2024). This normalisation may simplify the task.

Table 5.1 Log density of samples generated by compared methods, evaluated on the ground-truth target density of MoG-40. "True" indicates the log density of ground-truth samples from the target distribution, which serves as a reference. We only report the evaluation methods that can cover all the modes; see Figure 5.4 for a visualisation of samples generated by all baseline methods.

Method	True	FAB	iDEM	R-DiKL
$\log p_d(x)$	*-6.85*	-10.74	-8.33	**-7.21**

(2023), which allows us to visually examine the mass-covering property of our method. The baseline methods include R-KL with score matching (**R-KL-SM**) (Luo et al., 2023), R-KL with variational upper bound (**R-KL-VI**) (Zhang et al., 2019), a flow-based sampler (**FAB**) (Midgley et al., 2023), and a diffusion-based sampler (**iDEM**) (Akhound-Sadegh et al., 2024). Overall, our approach can cover all the modes and generate high-quality samples.

As shown in Figure 5.4, our approach achieves better sample quality than all baseline neural samplers. R-KL-based samplers struggle to capture the majority of modes due to its mode-seeking property. FAB captures all modes but exhibits heavy density connections between modes due to the limited expressivity of normalising flows. This is in contrast to our sampler which only requires using more flexible standard neural networks. As for iDEM, while its samples do not exhibit such density connections, they look noisy. This is because iDEM is a diffusion model which requires accurate score estimation across all noise levels (from the target distribution towards a pure Gaussian distribution) to generate accurate samples. However, the TSI score estimator in iDEM has high variance at larger noise levels. In contrast, our approach samples directly from a one-step generator $f_\theta(z)$ and only uses Gaussian convolution kernels to connect adjacent modes during training, allowing for a much smaller noise level and more manageable variance. Table 5.1 shows the log density of samples generated by these methods, confirming that our method achieves the best sampling performance.

5.5.2 Many-Body Particle Systems

Problem Setup

One important application of neural samplers is to generate samples from Boltzmann distributions, where the target distribution defines the probability density that a physical system will be in a certain state as a function of that state's energy and the temperature of the system. This type of neural sampler is also known as Boltzmann generator (Noé et al., 2019). For simplicity of notation, we will omit the temperature in the exponent and absorb it into the energy function.

We will evaluate the performance of our neural sampler $f_\theta(z)$ as a Boltzmann generator to generate samples from many-body particle systems, where the energy is defined over the pairwise distances between n particles. These systems can be defined in either internal or Cartesian coordinates. Note that for Cartesian coordinates, the energy of the system will remain invariant if we apply any of the rotation, reflection, translation, and permutation operations to the entire system. Formally, representing each configuration of the system by a matrix $X \in \mathbb{R}^{n\times d_x}$, our target distribution $p_d(X)$ is invariant to the product group of the Euclidean group and the Symmetric group of degree n, i.e. $G = \mathrm{E}(d) \times \mathbb{S}_n$.

This invariance presents a challenge to the training of neural samplers. Recall that our neural sampler learns a mapping $f_\theta : \mathcal{Z} \to \mathcal{X}$, where $\mathcal{X}$ represents the space of system configurations, and $\mathcal{Z}$ represents the latent space. If $\mathcal{X}$ includes configurations with symmetries but $\mathcal{Z}$ does not account for these symmetries, the network would need to model every equivariant configuration separately (e.g., the generator $f_\theta(z)$ would need to learn to assign the same density for all configurations in each equivariant class respect to G), leading to inefficient training and possibly poor generalisation.

On the other hand, one common approach is to parameterise the neural sampler $f_\theta(z)$ with an equivariant graph neural network (EGNN) (Hoogeboom et al., 2022; Satorras et al., 2021), ensuring that it is G-equivariant. In this case, the latent variable $Z \in \mathbb{R}^{n\times d_z}$ must have the same dimensionality as X (i.e., $d_z = d_x \coloneqq d$). We can show that as long as the distribution over the latent space $p_z(Z)$ is G-invariant, the model density $p_\theta(X)$ of a deep implicit model defined by Equation (5.21) is G-invariant.

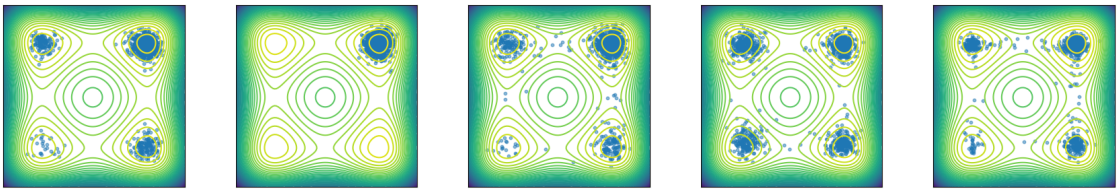
Proposition 5.5.1. *Let the neural sampler $f_\theta : \mathcal{Z} \to \mathcal{X}$ be a G-equivariant mapping. If the distribution $p_z(Z)$ over the latent space $\mathcal{Z}$ is G-invariant, then the model density:*

$$p_\theta(X) = \int \delta(X - f_\theta(Z))p_z(Z)dZ, \tag{5.29}$$

is also G-invariant.

The proof of Proposition 5.5.1 can be found in Appendix D.4.1. As a result, if we use the standard Gaussian distribution as the prior for the latent variables, our neural samplers will automatically enforce the desired invariance. This greatly reduces the challenges of training neural samplers in the Cartesian coordinate.

However, another challenge arises from the translation operation. As noted by Midgley et al. (2024), there does not exist a translation invariant probability measure in the Euclidean space. Therefore, following Akhound-Sadegh et al. (2024); Hoogeboom et al. (2022); Midgley et al. (2024); Satorras et al. (2021), we constrain both $\mathcal{X}$ and $\mathcal{Z}$ to be in the subspace of $\mathbb{R}^{n\times d}$ with



(a) Ground Truth (b) R-KL (c) FAB (d) iDEM (e) R-DiKL

Figure 5.5 2D marginal (1st and 3rd dimensions) of samples from MW-32. R-DiKL and FAB manage to find all the modes with correct weights. Note that iDEM finds all modes but with wrong weights. R-KL only captures one mode.

zero centre-of-mass (i.e., $X^\top 1 = Z^\top 1 = 0$). This allows us to embed the product group $G = \mathrm{E}(d) \times \mathbb{S}_n$ into an orthogonal group in the nd-dimensional space: $\mathrm{E}(d) \times \mathbb{S}_n \hookrightarrow \mathrm{O}(nd)$.

Having decided the neural network architecture for the neural sampler f_θ and tackled the translation invariance, we now turn to the scoring network $s_\phi(X_t) \approx \nabla_{X_t} \log p_\theta(X_t)$ for estimating the clean and noisy score functions of the neural sampler. According to Papamakarios et al. (2021, Lemma 2), *if G is a subgroup of the orthogonal group, then the gradient of a G-invariant function is G-equivariant.* Therefore, the score network $s_\phi(X_t)$ for the neural sampler that defines a G-invariant distribution should be G-equivariant. To achieve this, we model the score network $s_\phi(X_t)$ with an EGNN and train it within the zero-centred subspace, following Hoogeboom et al. (2022).

Note that when both the model and target density functions are G-invariant, the score of the log density ratio between the model and the target (i.e., $\nabla_{X_t} \log p_\theta(X_t) - \nabla_{X_t} \log p_d(X_t)$) which appears in the gradient of R-DiKL as shown in Equation (5.23) will also be G-equivariant. Therefore, we also need to ensure that the Monte Carlo estimator of MSI for the noisy target score $\nabla_{X_t} \log p_d(X_t)$ is G-equivariant. Recall that the MSI estimator is given by:

$$\nabla_{X_t} \log p_d(X_t) = \int (\alpha_t(X + \nabla_X \log p_d(X)) - X_t) p_d(X|X_t) dX. \tag{5.30}$$

Fortunately, this can be achieved by a broad class of Monte Carlo estimators under mild conditions, including IS and AIS estimators, with different choices of MCMC samplers for sampling the denoising posterior $p_d(X|X_t)$, such as MALA and HMC. Detailed derivations regarding the G-equivariance of Monte Carlo estimators can be found in Appendix D.4.2.

Many-Well-32 Potential in the Internal Coordinate

Midgley et al. (2023) introduced this Many-Well-32 (MW-32) potential in the internal coordinate by stacking 2D Double-Well potential 16 times, forming a distribution with 2^{16} modes in total. Moreover, these modes carry different weights. Therefore, this target distribution

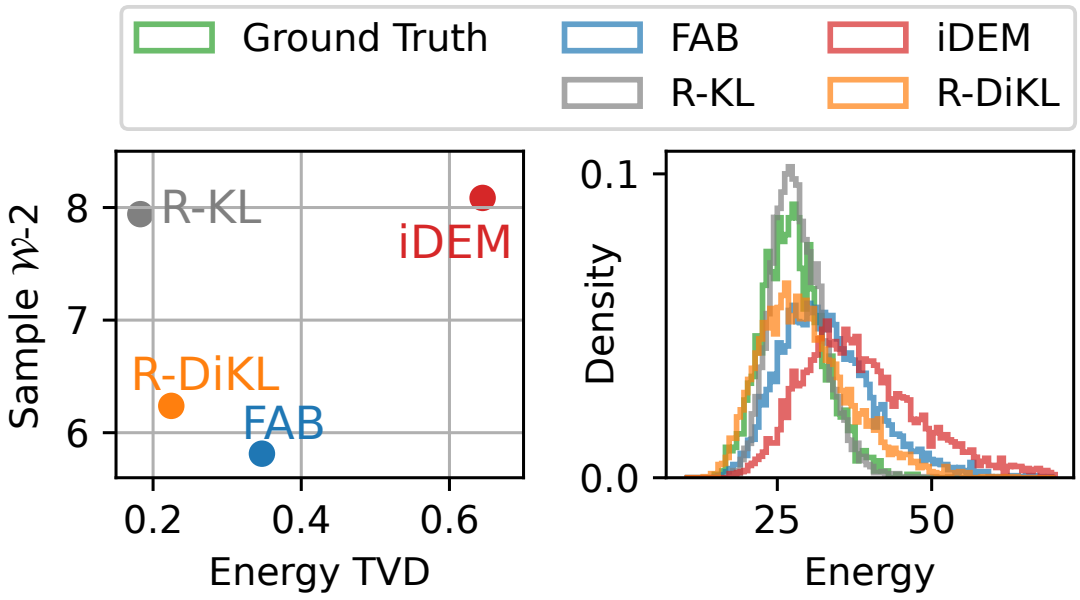
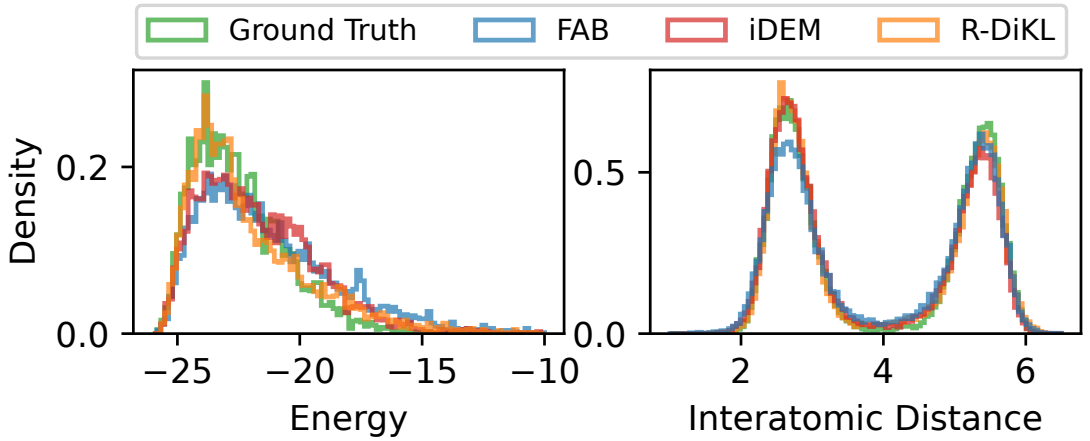


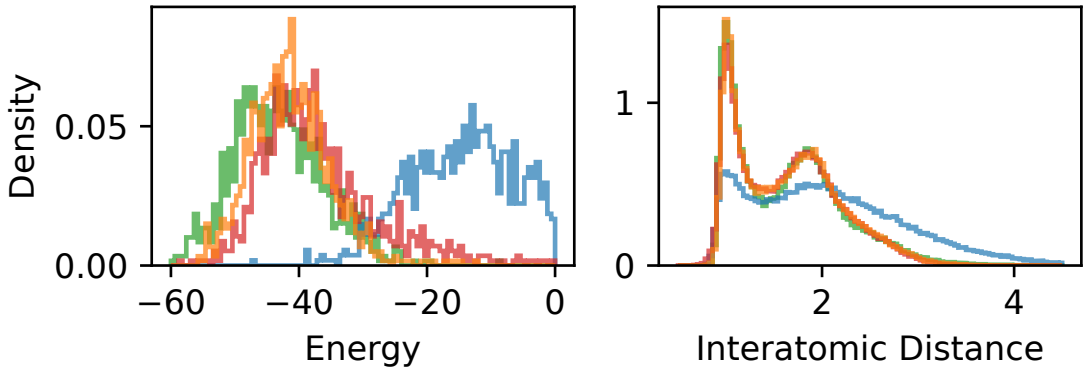
Figure 5.6 (Left) Wasserstein-2 ($\mathcal{W}$-2) distance of samples and total variation distance (TVD) for sample energy values on MW-32. R-DiKL and FAB clearly outperform iDEM and R-KL in this evaluation. (Right) Histogram of sample energy values. Our approach outperforms both FAB and iDEM. Note that although the R-KL yields better energy histogram, it collapses to only one mode, as shown in Figure 5.5.

can assess whether a neural sampler can successfully cover all modes and simultaneously capture their weights accurately. We compare **R-DiKL** with **FAB** and **iDEM**, which are the state-of-the-art flow-based and diffusion-based neural samplers on these target distributions. In addition, we include **R-KL** as a baseline for reference.

We report the Wasserstein-2 ($\mathcal{W}$-2) distances between samples generated by neural samplers and ground-truth samples. We also evaluate the energy values of the samples and report the total variation distance (TVD) between the distributions of energy values of ground-truth samples and samples generated by neural samplers. Note that both metrics have their limitations: $\mathcal{W}$-2 tends to be less sensitive to noisy samples, which can be particularly detrimental in some many-body particle systems. Conversely, the energy TVD is less sensitive to mode coverage. To provide a comprehensive evaluation, we plot both metrics together in Figure 5.6, which shows that overall our method and FAB clearly outperform iDEM and R-KL in this evaluation. We also visualise the samples along two selected marginal dimensions to assess mode coverage in Figure 5.5, showing that only our approach and FAB can find all modes with correct weights.

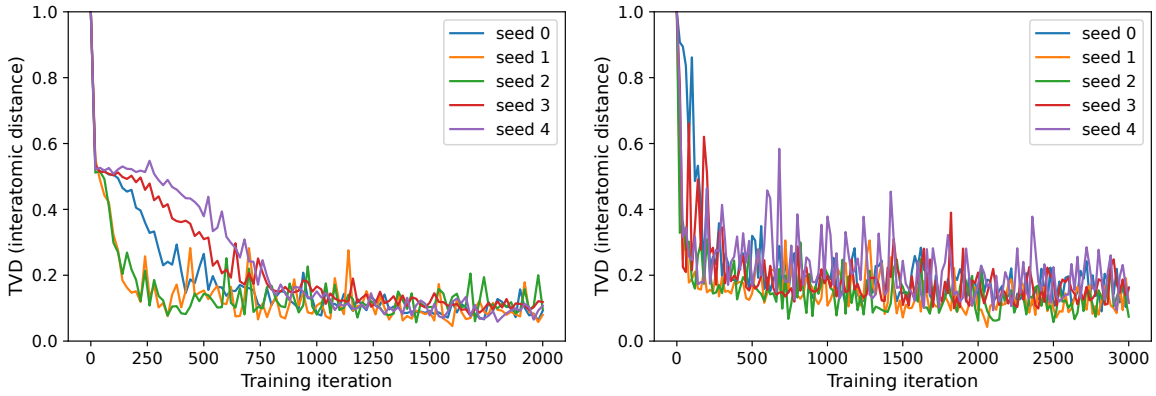


(a) DW-4



(b) LJ-13

Figure 5.7 Histograms of sample energy values and interatomic distances on DW-4 and LJ-13. R-DiKL achieves comparable performance to iDEM on both targets with only one function evaluation (NFE) at sampling time, while iDEM requires 1,000 NFEs.



(a) DW-4

(b) LJ-13

Figure 5.8 Total variation distance (TVD) for interatomic distance of R-DiKL samples as a function of the training iteration under different seeds on the DW-4 and LJ-13 potentials.

Table 5.2 Wasserstein-2 ($\mathcal{W}$-2) distance of samples, and total variation distances (TVDs) of sample energy values and interatomic distances for all compared methods on DW-4 and LJ-13. Each metric value is calculated using 5,000 samples and repeated ten times. The mean and standard deviation values are reported.

Method	Cartesian DW-4			Cartesian LJ-13		
	Sample $\mathcal{W}$-2	Energy TVD	Distance TVD	Sample $\mathcal{W}$-2	Energy TVD	Distance TVD
FAB	**1.554 ± 0.015**	0.224 ± 0.008	**0.097 ± 0.005**	4.938 ± 0.009	0.902 ± 0.010	0.252 ± 0.002
iDEM	1.593 ± 0.012	0.197 ± 0.010	0.103 ± 0.005	**4.172 ± 0.007**	0.306 ± 0.013	0.044 ± 0.001
R-DiKL	1.581 ± 0.026	**0.167 ± 0.012**	0.101 ± 0.006	4.233 ± 0.008	**0.239 ± 0.019**	**0.042 ± 0.002**

Double-Well-4 and Lennard-Jones-13 Potentials in the Cartesian Coordinate

Köhler et al. (2020) introduced Double-Well-4 (DW-4) and Lennard-Jones-13 (LJ-13) potentials in the Cartesian coordinate to access the performance of neural samplers trained on invariant target distributions. Specifically, the target energy is invariant to the product group $G = \mathrm{E}(d) \times \mathbb{S}_n$, where $n = 4, d = 2$ for DW-4 and $n = 13, d = 3$ for LJ-13. Note that LJ-13 is more complex than DW-4 in the sense that its energy landscape includes prohibitive regions that can destabilise training. However, DW-4 has its own challenges since it has two modes with different weights, and capturing these these two modes with correct weights can be a challenging task for neural samplers. We therefore evaluate our approach and other baselines on both of them for a comprehensively evaluation.

Again, we compare **R-DiKL** with state-of-the-art flow-based neural samplers **FAB** and diffusion-based neural samplers **iDEM**. We report Wasserstein-2 ($\mathcal{W}$-2) distance for samples, TVD for sample energy values and interatomic distance in Table 5.2. We also visualise histograms of sample energy values and interatomic distance in Figure 5.7, which shows that our method achieves competitive performance to both FAB and iDEM on DW-4, and notably outperforms FAB on LJ-13. The results demonstrate that our method achieves comparable or better performance than state-of-the-art flow-based and diffusion-based neural samplers.

Additionally, Figure 5.8 shows that the training process of our approach is robust across different random seeds in the sense that all runs converge to similar results in the end despite some fluctuations during training.

Training and Sampling Speed

We report the wall-clock time of both training and sampling for all compared neural samplers in Table 5.3. Notably, our method achieves faster training and sampling speed than both FAB and iDEM. FAB depends on a large normalising flow with limited expressiveness, which leads to significantly longer training times, particularly for complex tasks like LJ-13. In contrast, our approach maintains consistent training times across different tasks. On the other hand, iDEM

Table 5.3 Training and sampling wall-clock times for FAB, iDEM and our sampler, measured on a single NVIDIA A100 (80GB) GPU. We omit the sampling times for FAB on DW-4 and LJ13 as it is implemented in JAX with JIT compilation, making direct comparison with the other methods implemented in PyTorch not feasible. However, we expect FAB to have slightly slower sampling times than R-DiKL due to its larger flow architecture.

Phase	Potential	Method		
		FAB	iDEM	R-DiKL
Training	MW-32	3.5h	3.5h	**2.5h**
	DW-4	4.5h	4.5h	**0.9h**
	LJ-13	21.5h	**6.5h**	**6.5h**
Batch Sampling (1,000 samples)	MW-32	**0.01s**	7.2s	**0.01s**
	DW-4	–	2.6s	**0.01s**
	LJ-13	–	19.7s	**0.02s**

is a diffusion model that requires intensive computation for sampling by simulating the SDE denoising diffusion process, which is orders of magnitudes slower than our approach.

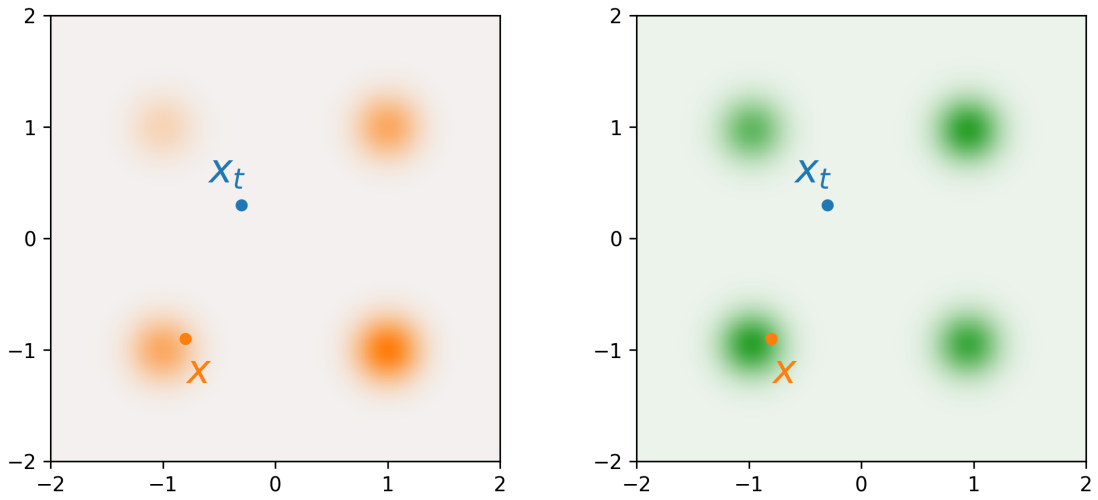
Summary of Results

Overall, our method achieves similar or better sampling performance compared to other types of state-of-the-art Boltzmann generators, FAB and iDEM, while having faster training and sampling speed. Furthermore, we would like to highlight that, unlike iDEM and FAB which use replay buffers to balance between exploration and exploitation, our approach does *not* rely on such replay buffer, thus offering a cleaner, more straightforward and easier-to-extend solution to Boltzmann generators.

5.6 Discussion

In this chapter, we presented a new paradigm for training neural samplers parameterised by deep implicit models using reverse diffusive KL (R-DiKL) divergence. We derived a tractable gradient estimator for practical model training which accounts for the symmetries of physical systems. The resulting training algorithm provides a simple yet efficient approach to training neural samplers with the mass-covering property for multi-modal target distributions. We demonstrated its effectiveness on both synthetic multi-modal target distributions and Boltzmann distributions for many-body particles systems. Our approach matched or even outperformed state-of-the-art flow-based and diffusion-based neural samplers with improved training and sampling efficiency and without relying on replay buffers.

The limitations of our approach and their implications are discussed below.



(a) $p_d(x)$ (b) $p_d(x|x_t)$

Figure 5.9 Visualisation of the density of an MoG-4 target distribution with unequal weights for the four Gaussian components. (a) Density heatmap of the target distribution $p_d(x)$, a clean sample x and a noisy sample x_t obtained with Gaussian convolution parameters $\alpha_t = 1, \sigma_t = 1$. (b) Density heatmap of the denoising posterior distribution $p_d(x|x_t)$.

1. The density $p_\theta(x)$ of our neural sampler is intractable, since we employ a deep implicit model with a non-linear and non-invertible generator $f_\theta(z)$. Compared to FAB, although we have a more flexible generator, we cannot use importance re-weighting to correct the potential bias of generated samples as we do not have access to the sample density.

2. Although our one-step generator $f_\theta(z)$ achieved significantly faster sampling speed than diffusion-based samplers, it has limited model flexibility compared to multi-step diffusion models such as iDEM. Consequently, it is more difficult for our model to handle target distributions with complicated energy landscapes such as LJ-55.

3. For the two many-body system potentials in the Cartesian coordinate (i.e., DW-4 and LJ-13), training can be unstable especially near convergence, and we had to employ some heuristic criteria to perform early stopping as discussed in Appendix D.5.

4. Denoising posterior sampling for the MSI estimator can be a bottleneck in our training procedure, since it requires running an MCMC sampler to generate samples from the denoising posterior distribution $p_d(x|x_t)$ at each training iteration. The MCMC sampling procedure may have a slow mixing speed for complicated target distributions $p_d(x)$. This is because when $p_d(x)$ has very disconnected modes, the resulting denoising posterior $p_d(x|x_t)$ may still exhibit a multi-modal nature; see Figure 5.9 for an example. Nevertheless, we note that it is not essential to have a perfect denoising posterior sampler,

since the posterior samples will get closer to the true targert as our model improves during training. We would also like to point out that most diffusion-based or diffusion-inspired samplers have similar issues, since there is no ground-truth samples available to train a denoiser or score network in the sampling setting.

5. Our approach focused on deriving a diffusive version of the KL divergence for training neural samplers with better mode-covering properties. In fact, there are other types of measures, such as Wasserstein metric or the more general Bregman divergence, which may inherently have better mode connectivity in theory. However, training neural samplers with those measures would require samples from the target distribution $p_d(x)$, which is unavailable in the sampling setting. One could obtain such samples by directly running MCMC samplers for the target distribution, but this would be very inefficient compared to our approach that only requires sampling from the denoising posterior $p_d(x|x_t)$, which allows for provably faster convergence speed for MCMC samplers (Huang et al., 2024; Vempala and Wibisono, 2019).

Chapter 6

Conclusions and Future Work

6.1 Thesis Summary

This thesis focused on designing new machine learning methods to bridge deep learning and probabilistic inference, demonstrating that the strengths of methods in these two fields can be combined to enhance each other. Chapter 3 and Chapter 4 investigated how to leverage probabilistic inference techniques to improve the data efficiency and identifiability of representation learning for deep neural networks on small datasets. Chapter 5 explored the possibility of leveraging generative deep learning approaches to enhance the efficiency of sampling-based probabilistic inference. The proposed new techniques were evaluated on real-world molecular modelling tasks, including molecular property prediction and molecular configuration sampling. Below, we provide a thorough summary of the work presented in each chapter.

Chapter 3 introduced a data-efficient meta-learning approach, ADKF-IFT, which utilises deep neural networks to extract useful feature representations shared across tasks and enables probabilistic inference for the adaptive task-specific regression heads, aiming to improve the robustness of deep neural networks on small datasets. The meta-learning problem was formulated within a novel bilevel optimisation framework, which provides a unifying framework that generalises previous methods for training deep kernel GPs, eliminating the overfitting and underfitting issues in prior work. The bilevel optimisation problem was efficiently solved using the implicit function theorem. We proposed a specific instantiation of the general ADKF-IFT framework, which adapts all GP base kernel parameters and meta-learns all neural network feature extractor parameters. This specific model was empirically evaluated on few-shot molecular property prediction and optimisation tasks, showcasing its superior performance on real-world chemical problems with small datasets compared to diverse baseline models.

Chapter 4 explored the theoretical properties of neural network representations learned across multiple tasks via a probabilistic approach, with a particular emphasis on recovering canonical feature representations that align with the underlying ground-truth data generating process from observed data. We demonstrated that linear identifiability could be achieved through a standard multi-task regression neural network (MTRN). Additionally, under certain assumptions of task structures, we established that point-wise identifiability may also be attainable with a multi-task linear causal model (MTLCM), providing a much stronger identifiability guarantee than prior work. Moreover, our approach enabled exact maximum marginal likelihood estimation, which simplifies and stabilises the training procedure compared to previous approaches. Notably, MTLCM is capable of distinguishing between causal and spurious latent variables as a side product. Furthermore, our model can be deployed to extract canonical feature representations at test time, since it does not require conditioning on the target variable during inference. The identifiability theory presented in this chapter were validated through experiments on both synthetic tasks and real-world molecular property prediction tasks.

Chapter 5 investigated a complementary perspective to Chapters 3 and 4, focusing on improving the efficiency of probabilistic inference with deep learning. Drawing inspiration from recent advancements in generative deep learning, we introduced a novel method for fitting deep generative models to unnormalised probability distributions using diffusion-based techniques but without ground-truth samples. Such deep generative models were referred to as neural samplers. We proposed to train neural samplers with the R-DiKL divergence, which encourages the models to explore all high density regions within the support of the target distribution. We derived a tractable gradient estimator for practical model training with this objective. We demonstrated that the proposed training algorithm enabled training neural samplers capable of generating accurate independent samples from both synthetic multi-modal distributions and real-world many-body particle systems in one step. Notably, the one-step neural samplers trained by our approach achieved comparable or superior performance than state-of-the-art flow-based and diffusion-based neural samplers with significantly faster training and inference speed and without relying on extra engineering tricks such as reply buffers.

6.2 Future Research Directions

This section outlines potential directions for future research, with the goal of inspiring future work that builds upon the new techniques and insights presented in this thesis. We begin by proposing concrete extensions for the individual work in Chapters 3-5, which can potentially further improve their performance and applicability. We then move on to a broader discussion of other promising research directions that explore the synergistic interplay between deep learning and probabilistic inference.

Individual Work

Although the ADKF-IFT approach presented in Chapter 3 achieved state-of-the-art performance on a variety of molecular property prediction and optimisation tasks, there are some aspects which can be improved. First, we used a single lengthscale parameter for all input features to the GP base kernel. Using automatic relevance determination (ARD) in the base kernel may improve the model performance, since ARD allows the GP model to automatically select relevant features for each individual task. The potential overfitting problems may be reduced by assuming a sparse prior over lengthscales or by learning a low-dimensional manifold for them. Second, we only tested one specific instantiation of the general ADKF-IFT framework, which adapts the GP base kernel parameters to each task and meta-learns all feature extractor parameters. Adapting the last few layers of the feature extractor in addition to the GP base kernel to each task may further improve the model performance, which can be achieved by allowing small deviations across tasks according to a meta-learned prior on the feature extractor parameters, e.g., as described in Chen et al. (2020). Third, we followed Patacchiola et al. (2020) to treat binary classification as ± 1 label regression, which can impede the model performance. Better few-shot classification performance may be achieved by adopting a more principled approximate inference strategy for few-shot GP classification, e.g., Pólya-Gamma data augmentation (Snell and Zemel, 2021) or Laplace approximation (Kim and Hospedales, 2021). Finally, it would be interesting to investigate how to inject domain expertise in drug discovery into the GP base kernel with hand-curated features and kernel combinations.

When modelling the underlying data generating process in Chapter 4, we assumed that the spurious correlation between the spurious latent factors and the target variable was due to an anti-causal relationship. However, this assumption does not capture all possible non-causal correlations between latent factors and the target variable. Therefore, it may be interesting for future work to incorporate the potential pairwise interactions between latent factors into the modelling assumptions. In addition, while our approach achieved state-of-the-art identifiability results on real-world molecular data, our preliminary investigation indicated that the resulting canonical feature representations did not lead to improved molecular property prediction performance. One possible explanation is that a random split of training and test tasks does not create a valid out-of-distribution evaluation setting. Consequently, models which rely on spurious anti-causal features may achieve similar or even better predictive performance on in-distribution test tasks. For future work, it would be valuable to design a meaningful out-of-distribution task split to better access the advantages of our proposed method in real-world problems. Moreover, it would be interesting to investigate the physical meanings for each of the recovered causal and spurious canonical latent factors, which may provide fresh insights for understanding the underlying biochemical mechanism of drugs.

For the one-step neural samplers trained with R-DiKL in Chapter 5, we provided a clean training paradigm without relying on any particular engineering tricks such as reply buffers. Having said that, for future work, it might be beneficial to employ a replay buffer similar to the ones used in FAB and iDEM to balance exploration and exploitation and stabilise training. Since we do not have access to the model density of our neural samplers, we could instead minimise the Fisher divergence evaluated at samples from a buffer. Moreover, we mentioned that posterior sampling could be a bottleneck of the proposed training algorithm, especially for complicated target distributions. If we do have access to some ground-truth samples $x \sim p_d(x)$ from the target distribution, we may use them as proposals for efficient posterior sampling, since the target distribution $p_d(x)$ and the denoising posterior $p_d(x|x_t)$ often exhibit significant overlap as shown in Figure 5.9. The IS weight for this importance sampling procedure is simply proportional to the Gaussian convolution kernel:

$$w_{\text{IS}}(x) = \frac{p_d(x|x_t)}{p_d(x)} \propto k(x_t|x). \tag{6.1}$$

This suggests a promising direction for future work: exploring how to combine a small amount of ground-truth target samples to accelerate the posterior sampling procedure when training neural samplers with R-DiKL. Furthermore, it is worth noting that our method requires training a new neural sampler from scratch for each target distribution. However, it practice, many target distributions share similar properties (e.g., Lennard-Jones potentials with varying numbers of particles). For future work, it would be valuable to explore methods that enable rapid adaptation of pre-trained neural samplers to related target distributions, potentially leveraging meta-learning and multi-task learning techniques from Chapter 3 and Chapter 4 to improve training efficiency and model generalisation.

Broader Discussion

More broadly, the synergies between deep learning and probabilistic inference open up exciting research opportunities beyond what has been explored in this thesis. For instance, while AlphaFold (Abramson et al., 2024; Jumper et al., 2021) provides a deep learning framework that enables highly accurate prediction of *static* protein folding structures from 1D amino acid sequences, it cannot sample all possible 3D structures from the underlying Boltzmann distributions. This limitation is significant, since access to diverse and accurate samples from the underlying Boltzmann distributions is essential for estimating key physical quantities that describe the macroscopic behaviours of protein dynamics. Therefore, a promising direction is to leverage probabilistic inference approaches to develop principled deep generative models that can efficiently produce accurate independent samples of protein configurations following the underlying Boltzmann distribution of any given 1D amino acid sequence.

Another interesting direction would be to investigate how deep generative models can be integrated into the probabilistic inference framework to enhance uncertainty quantification. For instance, in molecular property prediction, the predictive posterior distribution may not only be uncertain but also be multi-modal. Common approximate probabilistic inference approaches for BNNs, such as Laplace's approximation and variational inference, often underestimate the uncertainty and fail to capture most modes in the posterior predictive distribution. It would be valuable to explore how to leverage diffusion models to improve the quality of uncertainty estimates in such scenarios, since they are powerful tools for approximating multi-modal probability distributions and offer a natural way to incorporate conditioning variables (Dhariwal and Nichol, 2021; Ho and Salimans, 2021), which makes them well-suited for capturing the multi-modality and uncertainty of the posterior predictive distributions for deep neural networks.

One important caveat is that the significance of well-defined benchmarks and well-documented baselines is often overlooked. Arguably, the rapid progress in the computer vision community would not have been possible without standardised benchmarks such as ImageNet (Deng et al., 2009), which provided a transparent common ground for evaluating new models and quantifying advancements. Similarly, in order to make meaningful progress in bridging deep learning and probabilistic inference, it is crucial to design rigorous benchmarks that can accurately reflect how these methods would be applied in real-world applications and to develop reproducible baselines with open-source implementations and carefully tuned hyperparameters. The FS-Mol benchmark (Stanley et al., 2021) used in Chapter 3 serves as a strong example of such an effort, offering a well-curated benchmark for evaluating real-world few-shot molecular property prediction performance, which enables fair and comprehensive comparisons across a diverse set of representative baseline methods and encourages progress in developing data-efficient machine learning algorithms. Without such initiatives, it would be challenging to assess progress and identify truly promising methods.

References

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J. Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature*, 630(8016): 493–500, 2024.

Hagit Achdout, Anthony Aimon, Elad Bar-David, Haim Barr, Amir Ben-Shmuel, James Bennett, Vitaliy A. Bilenko, Vitaliy A. Bilenko, Melissa L. Boby, Bruce Borden, et al. Open science discovery of oral non-covalent SARS-CoV-2 main protease inhibitor therapeutics. *bioRxiv*, 2022.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Tara Akhound-Sadegh, Jarrid Rector-Brooks, Joey Bose, Sarthak Mittal, Pablo Lemos, Cheng-Hao Liu, Marcin Sendera, Siamak Ravanbakhsh, Gauthier Gidel, Yoshua Bengio, et al. Iterated denoising energy matching for sampling from Boltzmann densities. In *International Conference on Machine Learning*, 2024.

Michael S Albergo and Eric Vanden-Eijnden. NETS: A non-equilibrium transport sampler. *arXiv preprint arXiv:2410.02711*, 2024.

Anna Allen, Stratis Markou, Will Tebbutt, James Requeima, Wessel P Bruinsma, Tom R Andersson, Michael Herzog, Nicholas D Lane, Matthew Chantry, J Scott Hosking, et al. End-to-end data-driven weather prediction. *Nature*, pages 1–3, 2025.

Han Altae-Tran, Bharath Ramsundar, Aneesh S. Pappu, and Vijay S. Pande. Low data drug discovery with one-shot learning. *ACS Central Science*, 3:283 – 293, 2017.

Javier Antoran. *Scalable Bayesian inference in the era of deep learning: From Gaussian processes to deep neural networks*. PhD thesis, University of Cambridge, 2024.

Michael Arbel, Alex Matthews, and Arnaud Doucet. Annealed flow transport Monte Carlo. In *International Conference on Machine Learning*, pages 318–330. PMLR, 2021.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223. PMLR, 2017.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. Wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.

Fan Bao, Chongxuan Li, Jun Zhu, and Bo Zhang. Analytic-DPM: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. In *International Conference on Learning Representations*, 2022.

David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

David Barber and Christopher M Bishop. Ensemble learning in Bayesian neural networks. *Nato ASI Series F Computer and Systems Sciences*, 168:215–238, 1998.

Ilyes Batatia, David P Kovacs, Gregor Simm, Christoph Ortner, and Gábor Csányi. MACE: Higher order equivariant message passing neural networks for fast and accurate force fields. *Advances in neural information processing systems*, 35:11423–11436, 2022.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8): 1889–1900, 2000.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. GFlowNet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Technical Report*, 2023.

Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619(7970): 533–538, 2023.

Christopher M Bishop. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Christopher M Bishop and Hugh Bishop. *Deep learning: Foundations and concepts*. Springer Nature, 2023.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

Cristian Bodnar, Wessel P Bruinsma, Ana Lucic, Megan Stanley, Anna Allen, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan A Weyn, Haiyu Dong, et al. A foundation model for the earth system. *Nature*, pages 1–8, 2025.

Edwin V Bonilla, Kian Chai, and Christopher Williams. Multi-task Gaussian process prediction. *Advances in neural information processing systems*, 20, 2007.

John Bradshaw, Alexander G de G Matthews, and Zoubin Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*, 2017.

Jack Brady, Roland S Zimmermann, Yash Sharma, Bernhard Schölkopf, Julius Von Kügelgen, and Wieland Brendel. Provably learning object-centric representations. In *International Conference on Machine Learning*, pages 3038–3062. PMLR, 2023.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and deep locally connected networks on graphs. In *International Conference on Learning Representations*, 2014.

Roberto Calandra, Jan Peters, Carl E Rasmussen, and Marc Peter Deisenroth. Manifold Gaussian processes for regression. In *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016.

Rich Caruana. Multitask learning. *Machine learning*, 28:41–75, 1997.

Da Chen, Yuefeng Chen, Yuhong Li, Feng Mao, Yuan He, and Hui Xue. Self-supervised learning for few-shot image classification. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1745–1749. IEEE, 2021.

Junhua Chen, Lorenz Richter, Julius Berner, Denis Blessing, Gerhard Neumann, and Anima Anandkumar. Sequential controlled Langevin diffusions. In *International Conference on Learning Representations*, 2025a.

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, et al. WavLM: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1505–1518, 2022.

Wenlin Chen and Hong Ge. Neural characteristic activation analysis and geometric parameterization for ReLU networks. *Advances in Neural Information Processing Systems*, 37, 2024.

Wenlin Chen, Austin Tripp, and José Miguel Hernández-Lobato. Meta-learning adaptive deep kernel Gaussian processes for molecular property prediction. In *International Conference on Learning Representations*, 2023.

Wenlin Chen, Julien Horwood, Juyeon Heo, and José Miguel Hernández-Lobato. Leveraging task structures for improved identifiability in neural network representations. *Transactions on Machine Learning Research*, 2024a.

Wenlin Chen, Mingtian Zhang, Brooks Paige, José Miguel Hernández-Lobato, and David Barber. Diffusive Gibbs sampling. In *International Conference on Machine Learning*, pages 7731–7747. PMLR, 2024b.

Wenlong Chen, Wenlin Chen, Lapo Rastrelli, and Yingzhen Li. Your image is secretly the last frame of a pseudo video. In *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*, 2025b.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016.

Yutian Chen, Abram L Friesen, Feryal Behbahani, Arnaud Doucet, David Budden, Matthew Hoffman, and Nando de Freitas. Modular meta-learning with shrinkage. *Advances in Neural Information Processing Systems*, 33:2858–2869, 2020.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, 2014.

Ross M Clarke, Elre Talea Oldewage, and José Miguel Hernández-Lobato. Scalable one-pass optimisation of high-dimensional weight-update hyperparameters by implicit differentiation. In *International Conference on Learning Representations*, 2022.

Pierre Comon. Independent component analysis, a new concept? *Signal processing*, 36(3): 287–314, 1994.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in neural information processing systems*, 33:13260–13271, 2020.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux - effortless Bayesian deep learning. *Advances in neural information processing systems*, 34:20089–20103, 2021.

Valentin De Bortoli, Michael Hutchinson, Peter Wirnsberger, and Arnaud Doucet. Target score matching. *arXiv preprint arXiv:2402.08667*, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Lauro Langosco di Langosco, Vincent Fortuin, and Heiko Strathmann. Neural variational gradient descent. In *Fourth Symposium on Advances in Approximate Bayesian Inference*, 2022.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.

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

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987.

Alexandre Duval, Simon V Mathis, Chaitanya K Joshi, Victor Schmidt, Santiago Miret, Fragkiskos D Malliaros, Taco Cohen, Pietro Liò, Yoshua Bengio, and Michael Bronstein. A hitchhiker's guide to geometric GNNs for 3D atomic systems. *arXiv preprint arXiv:2312.07511*, 2023.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

Cian Eastwood, Alexander Robey, Shashank Singh, Julius Von Kügelgen, Hamed Hassani, George J Pappas, and Bernhard Schölkopf. Probable domain generalization via quantile risk minimization. *Advances in Neural Information Processing Systems*, 35:17340–17358, 2022.

Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

Alexander IJ Forrester, András Sóbester, and Andy J Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the royal society a: mathematical, physical and engineering sciences*, 463(2088):3251–3269, 2007.

Vincent Fortuin. Priors in Bayesian deep learning: A review. *International Statistical Review*, 90(3):563–591, 2022.

Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*. Elsevier, 2023.

Marco Fumero, Florian Wenzel, Luca Zancato, Alessandro Achille, Emanuele Rodolà, Stefano Soatto, Bernhard Schölkopf, and Francesco Locatello. Leveraging sparse and shared feature activations for disentangled representation learning. *Advances in Neural Information Processing Systems*, 36:27682–27698, 2023.

Yarin Gal. *Uncertainty in deep learning*. PhD thesis, University of Cambridge, 2016.

Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. In *International Conference on Learning Representations*, 2021.

Miguel García-Ortegón, Gregor NC Simm, Austin J Tripp, José Miguel Hernández-Lobato, Andreas Bender, and Sergio Bacallado. DOCKSTRING: easy molecular docking yields better benchmarks for ligand design. *Journal of chemical information and modeling*, 62 (15):3486–3502, 2022.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pages 1704–1713. PMLR, 2018.

Damien Garreau, Wittawat Jitkrittum, and Motonobu Kanagawa. Large sample analysis of the median heuristic. *arXiv preprint arXiv:1707.07269*, 2017.

Charles J Geyer and Elizabeth A Thompson. Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 90(431): 909–920, 1995.

Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521 (7553):452–459, 2015.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR, 2017.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323. PMLR, 2011.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks*, volume 2, pages 729–734, 2005.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Ulf Grenander and Michael I Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581, 1994.

Louis Grenioux, Maxence Noble, Marylou Gabrié, and Alain Oliviero Durmus. Stochastic localization via iterative posterior sampling. In *International Conference on Machine Learning*, pages 16337–16376. PMLR, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Zhichun Guo, Chuxu Zhang, Wenhao Yu, John Herr, Olaf Wiest, Meng Jiang, and Nitesh V Chawla. Few-shot graph learning for molecular property prediction. In *Proceedings of the web conference 2021*, pages 2559–2567, 2021.

John Gurland. On regularity conditions for maximum likelihood estimators. *Scandinavian Actuarial Journal*, 1954(1):71–76, 1954.

Johannes Hachmann, Roberto Olivares-Amaya, Sule Atahan-Evrenk, Carlos Amador-Bedolla, Roel S Sánchez-Carrera, Aryeh Gold-Parker, Leslie Vogt, Anna M Brockway, and Alán Aspuru-Guzik. The Harvard clean energy project: large-scale computational screening and design of organic photovoltaics on the world community grid. *The Journal of Physical Chemistry Letters*, 2(17):2241–2251, 2011.

Hermanni Hälvä, Sylvain Le Corff, Luc Lehéricy, Jonathan So, Yongjie Zhu, Elisabeth Gassiat, and Aapo Hyvarinen. Disentangling identifiable features from noisy data with structured nonlinear ica. *Advances in Neural Information Processing Systems*, 34:1624–1633, 2021.

Kam Hamidieh. Superconductivty Data. *UCI Machine Learning Repository*, 2018.

James Harrison, John Willes, and Jasper Snoek. Variational Bayesian last layers. In *International Conference on Learning Representations*, 2024.

W Keith Hastings. *Monte Carlo sampling methods using Markov chains and their applications*. Oxford University Press, 1970.

Jiajun He, Wenlin Chen, Mingtian Zhang, David Barber, and José Miguel Hernández-Lobato. Training neural samplers with reverse diffusive KL divergence. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Jeanny Herault and Christian Jutten. Space or time adaptive signal processing by neural network models. In *AIP conference proceedings*, volume 151, pages 206–211. American Institute of Physics, 1986.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International conference on machine learning*, pages 1861–1869. PMLR, 2015.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. Beta-VAE: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*, 2017.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

Geoffrey E Hinton and Russ R Salakhutdinov. Using deep belief nets to learn covariance kernels for Gaussian processes. *Advances in neural information processing systems*, 20, 2007.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

Matthew D Hoffman, Andrew Gelman, et al. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning*, pages 8867–8887. PMLR, 2022.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. HuBERT: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM transactions on audio, speech, and language processing*, 29:3451–3460, 2021.

Tianyang Hu, Zixiang Chen, Hanxi Sun, Jincheng Bai, Mao Ye, and Guang Cheng. Stein neural sampler. *arXiv preprint arXiv:1810.03545*, 2018.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020.

Xunpeng Huang, Hanze Dong, Yifan HAO, Yian Ma, and Tong Zhang. Reverse diffusion Monte Carlo. In *International Conference on Learning Representations*, 2024.

Ferenc Huszár. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Aapo Hyvärinen and Petteri Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural networks*, 12(3):429–439, 1999.

Aapo Hyvarinen, Hiroaki Sasaki, and Richard Turner. Nonlinear ICA using auxiliary variables and generalized contrastive learning. In *International Conference on Artificial Intelligence and Statistics*, pages 859–868. PMLR, 2019.

Leif Jacobson, James Stevenson, Farhad Ramezanghorbani, Steven Dajnowicz, and Karl Leswing. Leveraging multitask learning to improve the transferability of machine learned force fields. *ChemRxiv*, 2023.

Wengong Jin, Connor Coley, Regina Barzilay, and Tommi Jaakkola. Predicting organic reaction outcomes with Weisfeiler-Lehman network. *Advances in neural information processing systems*, 30, 2017.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

Robert O Jones. Density functional theory: Its origins, rise to prominence, and future. *Reviews of modern physics*, 87(3):897–923, 2015.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35: 26565–26577, 2022.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

Marc C Kennedy and Anthony O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.

Ilyes Khemakhem, Diederik Kingma, Ricardo Monti, and Aapo Hyvarinen. Variational autoencoders and nonlinear iCA: A unifying framework. In *International Conference on Artificial Intelligence and Statistics*, pages 2207–2217. PMLR, 2020a.

Ilyes Khemakhem, Ricardo Monti, Diederik Kingma, and Aapo Hyvarinen. ICE-BeeM: Identifiable conditional energy-based deep models based on nonlinear ICA. *Advances in Neural Information Processing Systems*, 33:12768–12778, 2020b.

Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang Dong Yoo. Edge-labeling graph neural network for few-shot learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11–20, 2019.

Minyoung Kim and Timothy Hospedales. Gaussian process meta few-shot classifier learning via linear discriminant laplace approximation. *arXiv preprint arXiv:2111.05392*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2013.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, pages 10215–10224, 2018.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. In *International Conference on Learning Representations*, 2023.

Bohdan Kivva, Goutham Rajendran, Pradeep Ravikumar, and Bryon Aragam. Identifiability of deep generative models without auxiliary information. *Advances in Neural Information Processing Systems*, 35:15687–15701, 2022.

Gregory R. Koch. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, 2015.

Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: exact likelihood generative learning for symmetric densities. In *International Conference on Machine Learning*, pages 5361–5370. PMLR, 2020.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pages 5815–5826. PMLR, 2021.

Sébastien Lachapelle, Pau Rodriguez, Yash Sharma, Katie E Everett, Rémi Le Priol, Alexandre Lacoste, and Simon Lacoste-Julien. Disentanglement via mechanism sparsity regularization: A new principle for nonlinear ICA. In *Conference on Causal Learning and Reasoning*, pages 428–484. PMLR, 2022.

Sébastien Lachapelle, Tristan Deleu, Divyat Mahajan, Ioannis Mitliagkas, Yoshua Bengio, Simon Lacoste-Julien, and Quentin Bertrand. Synergies between disentanglement and sparsity: Generalization and identifiability in multi-task learning. In *International Conference on Machine Learning*, pages 18171–18206. PMLR, 2023.

Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

J. Larsen, L.K. Hansen, C. Svarer, and M. Ohlsson. Design and regularization of neural networks: the optimal use of a validation set. In *Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*, pages 62–71, 1996.

Miguel Lázaro-Gredilla and Aníbal R Figueiras-Vidal. Marginalized neural network mixtures for large-scale regression. *IEEE transactions on neural networks*, 21(8):1345–1351, 2010.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Yin Tat Lee, Ruoqi Shen, and Kevin Tian. Structured logconcave sampling with a restricted Gaussian oracle. In *Conference on Learning Theory*, pages 2993–3050. PMLR, 2021.

Daniel Levy, Matt D. Hoffman, and Jascha Sohl-Dickstein. Generalizing Hamiltonian Monte Carlo with neural networks. In *International Conference on Learning Representations*, 2018.

Sarah Lewis, Tim Hempel, José Jiménez-Luna, Michael Gastegger, Yu Xie, Andrew Y. K. Foong, Victor García Satorras, Osama Abdin, Bastiaan S. Veeling, Iryna Zaporozhets, et al. Scalable emulation of protein equilibrium ensembles with generative deep learning. *bioRxiv*, pages 2024–12, 2024.

Yingzhen Li and Richard E. Turner. Gradient estimators for implicit models. In *International Conference on Learning Representations*, 2018.

Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *International Conference on Learning Representations*, 2016.

Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Feng Liu, Wenkai Xu, Jie Lu, and Danica J Sutherland. Meta two-sample testing: Learning kernels for testing with limited data. *Advances in Neural Information Processing Systems*, 34:5848–5860, 2021.

Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph normalizing flows. *Advances in Neural Information Processing Systems*, 32, 2019a.

Jeremiah Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *Advances in Neural Information Processing Systems*, 33:7498–7512, 2020.

Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sungju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. In *International Conference on Learning Representations*, 2019b.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*, pages 4114–4124. PMLR, 2019.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1552. PMLR, 2020.

Chaochao Lu, Yuhuai Wu, José Miguel Hernández-Lobato, and Bernhard Schölkopf. Invariant causal representation learning for out-of-distribution generalization. In *International Conference on Learning Representations*, 2022.

Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko. Scalable gradient-based tuning of continuous regularization hyperparameters. In *International Conference on Machine Learning*, pages 2952–2960. PMLR, 2016.

Weijian Luo, Boya Zhang, and Zhihua Zhang. Entropy-based training methods for scalable neural implicit samplers. *Advances in Neural Information Processing Systems*, 36, 2023.

Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-Instruct: A universal approach for transferring knowledge from pre-trained diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

Łukasz Maziarka, Tomasz Danel, Sławomir Mucha, Krzysztof Rataj, Jacek Tabor, and Stanisław Jastrzębski. Molecule attention transformer. *arXiv preprint arXiv:2002.08264*, 2020.

David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, et al. ChEMBL: towards direct deposition of bioassay data. *Nucleic acids research*, 47(D1): D930–D940, 2019.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

Laurence Midgley, Vincent Stimper, Javier Antorán, Emile Mathieu, Bernhard Schölkopf, and José Miguel Hernández-Lobato. SE (3) equivariant augmented coupling flows. *Advances in Neural Information Processing Systems*, 36, 2024.

Laurence Illing Midgley, Vincent Stimper, Gregor NC Simm, Bernhard Schölkopf, and José Miguel Hernández-Lobato. Flow annealed importance sampling bootstrap. In *International Conference on Learning Representations*, 2023.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013.

Erik G Miller, Nicholas E Matsakis, and Paul A Viola. Learning from one example through shared densities on transforms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 1, pages 464–471. IEEE, 2000.

Thomas Minka. Divergence measures and message passing. Technical report, Microsoft Research, 2005.

Hossein Mobahi and John W Fisher. On the link between Gaussian homotopy continuation and convex envelopes. In *Energy Minimization Methods in Computer Vision and Pattern Recognition: 10th International Conference, EMMCVPR 2015, Hong Kong, China, January 13-16, 2015. Proceedings 10*, pages 43–56. Springer, 2015.

J Harry Moore, Daniel J Cole, and Gabor Csanyi. Computing hydration free energies of small molecules with first principles accuracy. *arXiv preprint arXiv:2405.18171*, 2024.

Hiroshi Morioka, Hermanni Hälvä, and Aapo Hyvarinen. Independent innovation analysis for nonlinear vector autoregressive process. In *International Conference on Artificial Intelligence and Statistics*, pages 1549–1557. PMLR, 2021.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Radford M Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1996.

Radford M Neal. Annealed importance sampling. *Statistics and computing*, 11:125–139, 2001.

Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov chain Monte Carlo*, 2(11):2, 2011.

Kirill Neklyudov, Rob Brekelmans, Daniel Severo, and Alireza Makhzani. Action matching: Learning stochastic dynamics from samples. In *International Conference on Machine Learning*, pages 25858–25889. PMLR, 2023.

Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):eaaw1147, 2019.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

Nikolas Nusken, Francisco Vargas, Shreyas Padhy, and Denis Blessing. Transport meets variational inference: Controlled Monte Carlo diffusions. In *International Conference on Learning Representations*, 2024.

Sebastian W. Ober and Carl E. Rasmussen. Benchmarking the neural linear model for regression. In *Second Symposium on Advances in Approximate Bayesian Inference*, 2019.

Sebastian W Ober, Carl E Rasmussen, and Mark van der Wilk. The promises and pitfalls of deep kernel learning. In *Uncertainty in Artificial Intelligence*, pages 1206–1216. PMLR, 2021.

Zijing Ou, Mingtian Zhang, Andi Zhang, Tim Z. Xiao, Yingzhen Li, and David Barber. Improving probabilistic diffusion models with optimal diagonal covariance matching. In *The Thirteenth International Conference on Learning Representations*, 2025.

Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Eunbyung Park and Junier B Oliva. Meta-curvature. *Advances in neural information processing systems*, 32, 2019.

Massimiliano Patacchiola, Jack Turner, Elliot J Crowley, Michael O'Boyle, and Amos J Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. *Advances in Neural Information Processing Systems*, 33:16108–16118, 2020.

Massimiliano Patacchiola, John Bronskill, Aliaksandra Shysheya, Katja Hofmann, Sebastian Nowozin, and Richard Turner. Contextual squeeze-and-excitation for efficient few-shot image classification. *Advances in Neural Information Processing Systems*, 35:36680–36692, 2022.

Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. FourCastNet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *International Conference on Machine Learning*, pages 737–746. PMLR, 2016.

Ronan Perry, Julius Von Kügelgen, and Bernhard Schölkopf. Causal discovery in heterogeneous environments under the sparse mechanism shift hypothesis. *Advances in Neural Information Processing Systems*, 35:10904–10917, 2022.

Angus Phillips, Hai-Dang Dau, Michael John Hutchinson, Valentin De Bortoli, George Deligiannidis, and Arnaud Doucet. Particle denoising diffusion sampler. In *International Conference on Machine Learning*, pages 40688–40724. PMLR, 2024.

Matthias Poloczek, Jialei Wang, and Peter Frazier. Multi-information source optimization. *Advances in neural information processing systems*, 30, 2017.

Emilia Pompe, Chris Holmes, and Krzysztof Łatuszyński. A framework for adaptive MCMC targeting multimodal distributions. *The Annals of Statistics*, 48(5):2930–2952, 2020.

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion. In *International Conference on Learning Representations*, 2022.

Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32, 2019.

Liva Ralaivola, Sanjay J Swamidass, Hiroto Saigo, and Pierre Baldi. Graph kernels for chemical informatics. *Neural networks*, 18(8):1093–1110, 2005.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, January 2006.

Maria Reichenbach and P Morrison. The direction of time. *Physics Today*, 9(10):24–28, 1956.

Danilo Jimenez Rezende, George Papamakarios, Sébastien Racaniere, Michael Albergo, Gurtej Kanwar, Phiala Shanahan, and Kyle Cranmer. Normalizing flows on tori and spheres. In *International Conference on Machine Learning*, pages 8083–8092. PMLR, 2020.

Severi Rissanen, RuiKang OuYang, Jiajun He, Wenlin Chen, Markus Heinonen, Arno Solin, and José Miguel Hernández-Lobato. Progressive tempering sampler with diffusion. In *International Conference on Machine Learning*. PMLR, 2025.

Herbert E Robbins. An empirical Bayes approach to statistics. In *Breakthroughs in Statistics: Foundations and basic theory*, pages 388–394. Springer, 1992.

Gareth O Roberts and Osnat Stramer. Langevin diffusions and Metropolis-Hastings algorithms. *Methodology and computing in applied probability*, 4:337–357, 2002.

Gareth O Roberts and Richard L Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, pages 341–363, 1996.

Geoffrey Roeder, Luke Metz, and Durk Kingma. On linear identifiability of learned representations. In *International Conference on Machine Learning*, pages 9030–9039. PMLR, 2021.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

Mateo Rojas-Carulla, Bernhard Schölkopf, Richard Turner, and Jonas Peters. Invariant models for causal transfer learning. *The Journal of Machine Learning Research*, 19(1):1309–1342, 2018.

Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.

Warren S. Sarle. Stopped training and other remedies for overfitting. In *Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics*, pages 352–360, 1995.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In *International Conference on Machine Learning*, pages 9323–9332. PMLR, 2021.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Bernhard Schölkopf. Causality for machine learning. In *Probabilistic and Causal Inference: The Works of Judea Pearl*, page 765–804. Association for Computing Machinery, New York, NY, USA, 1 edition, 2022. ISBN 9781450395861.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

Kristof T Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R Müller, and Alexandre Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8(1):1–8, 2017.

Marcin Sendera, Minsu Kim, Sarthak Mittal, Pablo Lemos, Luca Scimeca, Jarrid Rector-Brooks, Alexandre Adam, Yoshua Bengio, and Nikolay Malkin. Improved off-policy training of diffusion samplers. *Advances in Neural Information Processing Systems*, 37: 81016–81045, 2024.

Jiaxin Shi, Shengyang Sun, and Jun Zhu. Kernel implicit variational inference. In *International Conference on Learning Representations*, 2018.

Martin Simonovsky and Nikos Komodakis. GraphVAE: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27*, pages 412–422. Springer, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2017.

Jake Snell and Richard Zemel. Bayesian few-shot classification with one-vs-each Pólya-Gamma augmented Gaussian processes. In *International Conference on Learning Representations*, 2021.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b.

Megan Stanley, John F Bronskill, Krzysztof Maziarz, Hubert Misztela, Jessica Lanini, Marwin Segler, Nadine Schneider, and Marc Brockschmidt. FS-Mol: A few-shot learning dataset of molecules. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.

Jingtong Sun, Julius Berner, Lorenz Richter, Marius Zeinhofer, Johannes Müller, Kamyar Azizzadenesheli, and Anima Anandkumar. Dynamical measure transport and neural PDE solvers for sampling. *arXiv preprint arXiv:2407.07873*, 2024.

Nikola Surjanovic, Saifuddin Syed, Alexandre Bouchard-Côté, and Trevor Campbell. Parallel tempering with a variational reference. *Advances in Neural Information Processing Systems*, 35:565–577, 2022.

Robert H Swendsen and Jian-Sheng Wang. Replica Monte Carlo simulation of spin-glasses. *Physical review letters*, 57(21):2607, 1986.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task Bayesian optimization. *Advances in neural information processing systems*, 26, 2013.

Saifuddin Syed, Alexandre Bouchard-Côté, George Deligiannidis, and Arnaud Doucet. Non-reversible parallel tempering: a scalable highly parallel MCMC scheme. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(2):321–350, 2022.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *European Conference on Computer Vision*, pages 266–282. Springer, 2020.

Petru Tighineanu, Kathrin Skubch, Paul Baireuther, Attila Reiss, Felix Berkenkamp, and Julia Vinogradska. Transfer learning with Gaussian processes for Bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 6152–6181. PMLR, 2022.

Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 61(3):611–622, 1999.

Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial intelligence and statistics*, pages 567–574. PMLR, 2009.

Marcin Tomczak, Siddharth Swaroop, Andrew Foong, and Richard Turner. Collapsed variational bounds for Bayesian neural networks. *Advances in Neural Information Processing Systems*, 34:25412–25426, 2021.

Prudencio Tossou, Basile Dura, Francois Laviolette, Mario Marchand, and Alexandre Lacoste. Adaptive deep kernel learning. *arXiv preprint arXiv:1905.12131*, 2019.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2020.

Austin Tripp, Wenlin Chen, and José Miguel Hernández-Lobato. An evaluation framework for the objective functions of de novo drug design benchmarks. In *ICLR 2022 Workshop on Machine Learning for Drug Discovery (MLDD)*, 2022.

Austin Tripp, Sergio Bacallado, Sukriti Singh, and José Miguel Hernández-Lobato. Tanimoto random features for scalable molecular machine learning. *Advances in Neural Information Processing Systems*, 36:33656–33686, 2023.

Brian Trippe and Richard Turner. Overpruning in variational Bayesian neural networks. *arXiv preprint arXiv:1801.06230*, 2018.

Richard Eric Turner and Maneesh Sahani. Two problems with variational expectation maximisation for time-series models. *Bayesian time series models*, 2011.

Joost van Amersfoort, Lewis Smith, Andrew Jesson, Oscar Key, and Yarin Gal. On feature collapse and deep kernel learning for single forward pass uncertainty. *arXiv preprint arXiv:2102.11409*, 2021.

Francisco Vargas, Will Sussman Grathwohl, and Arnaud Doucet. Denoising diffusion samplers. In *International Conference on Learning Representations*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Santosh Vempala and Andre Wibisono. Rapid convergence of the unadjusted Langevin algorithm: Isoperimetry suffices. *Advances in neural information processing systems*, 32, 2019.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

P Walters. We need better benchmarks for machine learning in drug discovery. *Practical Cheminformatics*, 2023.

Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023a.

Yaqing Wang, Abulikemu Abuduweili, Quanming Yao, and Dejing Dou. Property-aware relation networks for few-shot molecular property prediction. *Advances in Neural Information Processing Systems*, 34:17441–17454, 2021.

Yuanhao Wang, Guodong Zhang, and Jimmy Ba. On solving minimax optimization locally: A follow-the-ridge approach. In *International Conference on Learning Representations*, 2020.

Zeyu Wang, Tianyi Jiang, Yao Lu, Xiaoze Bao, Shanqing Yu, Bin Wei, and Qi Xuan. Knowledge-enhanced relation graph and task sampling for few-shot molecular property prediction. *arXiv preprint arXiv:2405.15544*, 2024.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. ProlificDreamer: High-fidelity and diverse text-to-3D generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36, 2023b.

Joe Watson, Jihao Andreas Lin, Pascal Klink, Joni Pajarinen, and Jan Peters. Latent derivative Bayesian last layer networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1198–1206. PMLR, 2021.

Oren F. Webb, Tommy J. Phelps, Paul R. Bienkowski, Philip M. Digrazia, David C. White, and Gary S. Sayler. Enzyme nomenclature, 1992.

David Weininger. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

David Weininger. SMILES. 3. depict. graphical depiction of chemical structures. *Journal of chemical information and computer sciences*, 30(3):237–243, 1990.

David Weininger, Arthur Weininger, and Joseph L Weininger. SMILES. 2. algorithm for generation of unique SMILES notation. *Journal of chemical information and computer sciences*, 29(2):97–101, 1989.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *International Conference on Machine Learning*, pages 681–688, 2011.

Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.

Matthew Willetts and Brooks Paige. I don't need u: Identifiable non-linear ICA without side information. *arXiv preprint arXiv:2106.05238*, 2021.

Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. *Advances in Neural Information Processing Systems*, 2016a.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016b.

Martin Wistuba and Josif Grabocka. Few-shot Bayesian optimization with deep kernel surrogates. In *International Conference on Learning Representations*, 2021.

Hao Wu, Jonas Köhler, and Frank Noé. Stochastic normalizing flows. *Advances in Neural Information Processing Systems*, 33:5933–5944, 2020.

Wen Wu, Wenlin Chen, Chao Zhang, and Phil Woodland. Modelling variability in human annotator simulation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 1139–1157, 2024.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Sirui Xie, Zhisheng Xiao, Diederik P Kingma, Tingbo Hou, Ying Nian Wu, Kevin Patrick Murphy, Tim Salimans, Ben Poole, and Ruiqi Gao. EM distillation for one-step diffusion models. *Advances in Neural Information Processing Systems*, 37, 2024.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Mingzhang Yin and Mingyuan Zhou. Semi-implicit variational inference. In *International Conference on Machine Learning*, pages 5660–5669. PMLR, 2018.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Dinghuai Zhang, Ricky TQ Chen, Cheng-Hao Liu, Aaron Courville, and Yoshua Bengio. Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. In *International Conference on Learning Representations*, 2024.

Miao Zhang, Steven W Su, Shirui Pan, Xiaojun Chang, Ehsan M Abbasnejad, and Reza Haffari. iDARTS: Differentiable architecture search with stochastic implicit gradients. In *International Conference on Machine Learning*, pages 12557–12566. PMLR, 2021.

Mingtian Zhang, Thomas Bird, Raza Habib, Tianlin Xu, and David Barber. Variational f-divergence minimization. *arXiv preprint arXiv:1907.11891*, 2019.

Mingtian Zhang, Peter Hayes, Thomas Bird, Raza Habib, and David Barber. Spread divergence. In *International Conference on Machine Learning*, pages 11106–11116. PMLR, 2020.

Mingtian Zhang, Alex Hawkins-Hooker, Brooks Paige, and David Barber. Moment matching denoising Gibbs sampling. *Advances in Neural Information Processing Systems*, 36:23590–23606, 2023a.

Mingtian Zhang, Wenlin Chen, Jiajun He, Zijing Ou, José Miguel Hernández-Lobato, Bernhard Schölkopf, and David Barber. Towards training one-step diffusion models without distillation. In *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*, 2025.

Qinsheng Zhang and Yongxin Chen. Path integral sampler: A stochastic control approach for sampling. In *International Conference on Learning Representations*, 2022.

Xuan Zhang, Limei Wang, Jacob Helwig, Youzhi Luo, Cong Fu, Yaochen Xie, Meng Liu, Yuchao Lin, Zhao Xu, Keqiang Yan, et al. Artificial intelligence for science in quantum, atomistic, and continuum systems. *arXiv preprint arXiv:2307.08423*, 2023b.

Appendix A

Supplementary Material for Chapter 2

A.1 Derivation of Denoising Score Identity

Proposition 2.3.1 (Denoising Score Identity). *For any convolution kernel $k_t(x_t|x)$, we have*

$$\nabla_{x_t} \log p_{d,t}(x_t) = \int \nabla_{x_t} \log k_t(x_t|x) p_d(x|x_t)\, \mathrm{d}x, \tag{A.1}$$

where $p_d(x|x_t) \propto k_t(x_t|x)p_d(x)$ is the denoising posterior distribution. For $k_t(x_t|x) = \mathcal{N}(x_t|\alpha_t x, \sigma_t^2 I)$ as defined in Equation (2.87), this recovers Tweedie's formula.

Proof. It follows that

$$\begin{aligned}
\nabla_{x_t} \log p_{d,t}(x_t) &= \frac{\nabla_{x_t} p_{d,t}(x_t)}{p_{d,t}(x_t)} && \text{(A.2)}\\
&= \frac{\nabla_{x_t} \int k_t(x_t|x) p_d(x)\, \mathrm{d}x}{p_{d,t}(x_t)} && \text{(A.3)}\\
&= \frac{\int \nabla_{x_t} k_t(x_t|x) p_d(x)\, \mathrm{d}x}{p_{d,t}(x_t)} && \text{(A.4)}\\
&= \frac{\int \nabla_{x_t} \log k_t(x_t|x) k_t(x_t|x) p_d(x)\, \mathrm{d}x}{p_{d,t}(x_t)} && \text{(A.5)}\\
&= \int \nabla_{x_t} \log k_t(x_t|x) \frac{k_t(x_t|x) p_d(x)\, \mathrm{d}x}{p_{d,t}(x_t)}\, \mathrm{d}x && \text{(A.6)}\\
&= \int \nabla_{x_t} \log k_t(x_t|x) p_d(x|x_t)\, \mathrm{d}x. && \text{(A.7)}
\end{aligned}$$

For $k_t(x_t|x) = \mathcal{N}(x_t|\alpha_t x, \sigma_t^2 I)$, we have

$$\nabla_{x_t} \log k_t(x_t|x) = \frac{\alpha_t x - x_t}{\sigma_t^2}, \tag{A.8}$$

which recovers the Tweedie's formula:

$$\nabla_{x_t} \log p_{d,t}(x_t) = \int \nabla_{x_t} \log k_t(x_t|x) p_d(x|x_t) \, \mathrm{d}x \tag{A.9}$$

$$= \frac{1}{\sigma_t^2} \left(\alpha_t \int x p_d(x|x_t) \, \mathrm{d}x - x_t \right). \tag{A.10}$$

This completes the proof. Note that the same argument can be used to derive DSI and Tweedie's formula for the noisy model score $\nabla_{x_t} \log p_{\theta,t}(x_t)$. □

A.2 Derivation of Denoising Score Matching

Without loss of generality, we consider a single time step t and set the weighting function to $\lambda(t) = 1$. We start from the score matching (SM) loss (Hyvärinen, 2005), expand its L2 norm, ignore the constants that are independent of θ, and apply the DSI from Proposition 2.3.1:

$$\mathcal{L}_{\text{SM}}(\theta) = \mathbb{E}_{p_{d,t}(x_t)}[\|s_\theta(x_t, t) - \nabla_{x_t} \log p_{d,t}(x_t)\|^2] \tag{A.11}$$

$$= \int \|s_\theta(x_t, t) - \nabla_{x_t} \log p_{d,t}(x_t)\|^2 p_{d,t}(x_t) \, \mathrm{d}x_t \tag{A.12}$$

$$= \int \|s_\theta(x_t, t)\|^2 p_{d,t}(x_t) \, \mathrm{d}x_t$$

$$- 2 \int s_\theta(x_t, t)^\top \nabla_{x_t} \log p_{d,t}(x_t) p_{d,t}(x_t) \, \mathrm{d}x_t + \text{const.} \tag{A.13}$$

$$= \int \|s_\theta(x_t, t)\|^2 p_{d,t}(x_t) \, \mathrm{d}x_t$$

$$- 2 \iint s_\theta(x_t, t)^\top \nabla_{x_t} \log k_t(x_t|x) p_d(x|x_t) p_{d,t}(x_t) \, \mathrm{d}x \, \mathrm{d}x_t + \text{const.} \tag{A.14}$$

$$= \iint \|s_\theta(x_t, t)\|^2 k_t(x_t|x) p_d(x) \, \mathrm{d}x \, \mathrm{d}x_t$$

$$- 2 \iint s_\theta(x_t, t)^\top \nabla_{x_t} \log k_t(x_t|x) k_t(x_t|x) p_d(x) \, \mathrm{d}x \, \mathrm{d}x_t + \text{const.} \tag{A.15}$$

$$= \iint \left(\|s_\theta(x_t, t)\|^2 + \|\nabla_{x_t} \log k_t(x_t|x)\|^2 - 2 s_\theta(x_t, t)^\top \nabla_{x_t} \log k_t(x_t|x) \right)$$

$$k_t(x_t|x) p_d(x) \, \mathrm{d}x \, \mathrm{d}x_t + \text{const.} \tag{A.16}$$

$$= \iint \|s_\theta(x_t, t) - \nabla_{x_t} \log k_t(x_t|x)\|^2 k_t(x_t|x) p_d(x) \, \mathrm{d}x \, \mathrm{d}x_t + \text{const.} \tag{A.17}$$

$$= \mathbb{E}_{p_d(x) k_t(x_t|x)}[\|s_\theta(x_t, t) - \nabla_{x_t} \log k_t(x_t|x)\|^2] + \text{const.} \tag{A.18}$$

$$= \mathcal{L}_{\text{DSM}}(\theta) + \text{const.} \tag{A.19}$$

This shows the equivalence between score matching and denoising score matching.

Appendix B

Supplementary Material for Chapter 3

B.1 Cauchy's Implicit Function Theorem

We state Cauchy's Implicit Function Theorem (IFT) in the context of ADKF-IFT.

Theorem B.1.1 (Implicit Function Theorem)**.** *Let $\mathcal{T}'$ be any given task. Suppose for some ψ'_{meta} and ψ'_{adapt} that $\frac{\partial \mathcal{L}_T(\psi_{meta},\psi_{adapt},\mathcal{S}_{\mathcal{T}'})}{\partial \psi_{adapt}}\Big|_{\psi'_{meta},\psi'_{adapt}} = 0$. Suppose that $\frac{\partial \mathcal{L}_T}{\partial \psi_{adapt}}(\psi_{meta},\psi_{adapt},\mathcal{S}_{\mathcal{T}'}) : \Psi_{meta} \times \Psi_{adapt} \to \Psi_{adapt}$ is a continuously differentiable function with respect to ψ_{meta} and ψ_{adapt}, and the Hessian $\frac{\partial^2 \mathcal{L}_T(\psi_{meta},\psi_{adapt},\mathcal{S}_{\mathcal{T}'})}{\partial \psi_{adapt}\partial \psi_{adapt}^\top}\Big|_{\psi'_{meta},\psi'_{adapt}}$ is invertible. Then, there exists an open set $U \in \Psi_{meta}$ containing ψ'_{meta} and a function $\psi^*_{adapt}(\psi_{meta},\mathcal{S}_{\mathcal{T}'}) : \Psi_{meta} \to \Psi_{adapt}$, such that $\psi'_{adapt} = \psi^*_{adapt}(\psi'_{meta},\mathcal{S}_{\mathcal{T}'})$ and $\frac{\partial \mathcal{L}_T(\psi_{meta},\psi_{adapt},\mathcal{S}_{\mathcal{T}'})}{\partial \psi_{adapt}}\Big|_{\psi''_{meta},\psi^*_{adapt}(\psi''_{meta},\mathcal{S}_{\mathcal{T}'})} = 0,\ \forall \psi''_{meta} \in U$. Moreover, the partial derivative of $\psi^*_{adapt}(\psi_{meta},\mathcal{S}_{\mathcal{T}'})$ with respect to ψ_{meta} for any $\psi''_{meta} \in U$ is given by*

$$\begin{aligned}
&\frac{\partial \psi^*_{adapt}(\psi_{meta},\mathcal{S}_{\mathcal{T}'})}{\partial \psi_{meta}}\Big|_{\psi''_{meta}} \\
&\quad = -\left(\frac{\partial^2 \mathcal{L}_T(\psi_{meta},\psi_{adapt},\mathcal{S}_{\mathcal{T}'})}{\partial \psi_{adapt}\partial \psi_{adapt}^\top}\right)^{-1} \frac{\partial^2 \mathcal{L}_T(\psi_{meta},\psi_{adapt},\mathcal{S}_{\mathcal{T}'})}{\partial \psi_{adapt}\partial \psi_{meta}^\top}\Big|_{\psi''_{meta},\psi^*_{adapt}(\psi''_{meta},\mathcal{S}_{\mathcal{T}'})}.
\end{aligned} \tag{B.1}$$

B.2 Configurations of ADKF-IFT

For all experiments, we use the specific instantiation of ADKF-IFT from Section 3.3.4.

We solve the inner optimisation problem (3.2) using the L-BFGS optimiser (Liu and Nocedal, 1989), since L-BFGS is the default choice for optimising base kernel parameters in the GP

literature. For the outer optimisation problem (3.1), we approximate the expected hypergradient over $p(\mathcal{T})$ by averaging the hypergradients over a batch of K randomly sampled training tasks at each step, and update the meta-learned parameters ψ_{meta} with the averaged hypergradient using the Adam optimiser (Kingma and Ba, 2014) with a learning rate of 10^{-3} for MoleculeNet and 10^{-4} for FS-Mol. We set $K = 10$ for MoleculeNet and $K = 16$ for FS-Mol. For all experiments on FS-Mol, we evaluate the performance of our model on a small set of validation tasks during meta-training and use early stopping (Prechelt, 1998) to avoid overfitting of the meta-learned parameters ψ_{meta}.

We use zero mean function and set Matérn-$5/2$ without automatic relevance determination (ARD) (Neal, 1996) as the base kernel in ADKF-IFT, since the typical sizes of the support sets in few-shot learning are too small to adjust a relatively large number of ARD lengthscales in ADKF-IFT. The lengthscale in the base kernel of ADKF-IFT is initialised using the median heuristic (Garreau et al., 2017) for each task, with a log-normal prior centred at the initialisation. Following Patacchiola et al. (2020), we treat binary classification as ± 1 label regression for ADKF-IFT.

B.3 Configurations of All Baselines on FS-Mol

Single-Task Methods

Single-task methods (RF, kNN, GP-ST, GNN-ST, and DKL) are trained separately on the support set of each test task, without leveraging the knowledge contained in the training tasks. The implementations of RF, kNN, and GNN-ST are taken from Stanley et al. (2021). RF, kNN, and GP-ST operates on top of manually curated features obtained using RDKit. RF and kNN use extended connectivity fingerprint (Rogers and Hahn, 2010) (count-based fingerprint with radius 2 and size 2,048) and phys-chem descriptors (with size 42). GP-ST uses fingerprint (with radius 2 and 2,048 bits based on count simulation). DKL operates on top of a combination of extended connectivity fingerprint (Rogers and Hahn, 2010) (count-based fingerprint with radius 2 and size 2,048) and features extracted by a GNN. The base kernel used in DKL is the same as that used in ADKF-IFT. DKL is trained for 50 epochs on the support set of each test task. Hyperparameter search configurations for these methods are based on the extensive industrial experience from the authors of Stanley et al. (2021). GNN-ST uses a GNN with a hidden dimension of 128 and a gated readout function (Gilmer et al., 2017), considering ~ 30 hyperparameter search configurations.

Multi-Task Pretraining

The implementation of GNN-MT is taken from Stanley et al. (2021). GNN-MT shares a GNN with a hidden dimension of 128 using principal neighbourhood message aggregation (Corso et al., 2020) across tasks, and uses a task-specific gated readout function Gilmer et al. (2017) and an MLP with one hidden layer on top for each individual task. The model is trained on the support sets of all training tasks with early stopping based on the performance on the validation tasks. The task-specific components of the model are fine-tuned for each test task.

Self-Supervised Pretraining

The implementation of MAT is taken from Stanley et al. (2021). We use the official pretrained model parameters (Maziarka et al., 2020), which is pretrained on 2 million molecules sampled from the ZINC15 dataset (Sterling and Irwin, 2015). We fine-tuned it for each test task with hyperparameter search and early stopping based on 20% of the support set for each task.

Meta-Learning Methods

Meta-learning methods (PAR, ProtoNet, GNN-MAML, CNP, DKT, and ADKF-IFT) enable knowledge transfer among related small datasets. The implementations of ProtoNet and GNN-MAML are taken from Stanley et al. (2021). The implementation of PAR is taken from its official implementation and integrated into the FS-Mol training and evaluation pipeline. PAR, ProtoNet, CNP, DKT, and ADKF-IFT operate on top of a combination of extended connectivity fingerprint (Rogers and Hahn, 2010) (count-based fingerprint with radius 2 and size 2,048) and features extracted by a GNN. The GNN feature extractor architecture used for DKL, PAR, CNP, DKT, and ADKF-IFT is the same as that used for ProtoNet, GNN-MAML, GNN-ST, and GNN-MT in Stanley et al. (2021), with the size of the feature representation being tuned on the validation tasks. The base kernel used in DKT is the same as that used in ADKF-IFT.

B.4 Further Comparisons Between DKT and ADKF-IFT

Figure B.1 and Figure B.2 visualise the distributions of the optimal ADKF-IFT base kernel parameters θ against the optimal DKT base kernel parameters on all FS-Mol test tasks. First, it can be seen that the optimal base kernel parameters vary across tasks in ADKF-IFT, demonstrating the importance of adapting these hyperparameters to each task. Moreover, ADKF-IFT generally has larger signal amplitude and smaller likelihood noise than DKT, achieving a significantly better signal-to-noise ratio than DKT in all settings. This confirms that ADKF-IFT provides a more informative prediction than DKT.

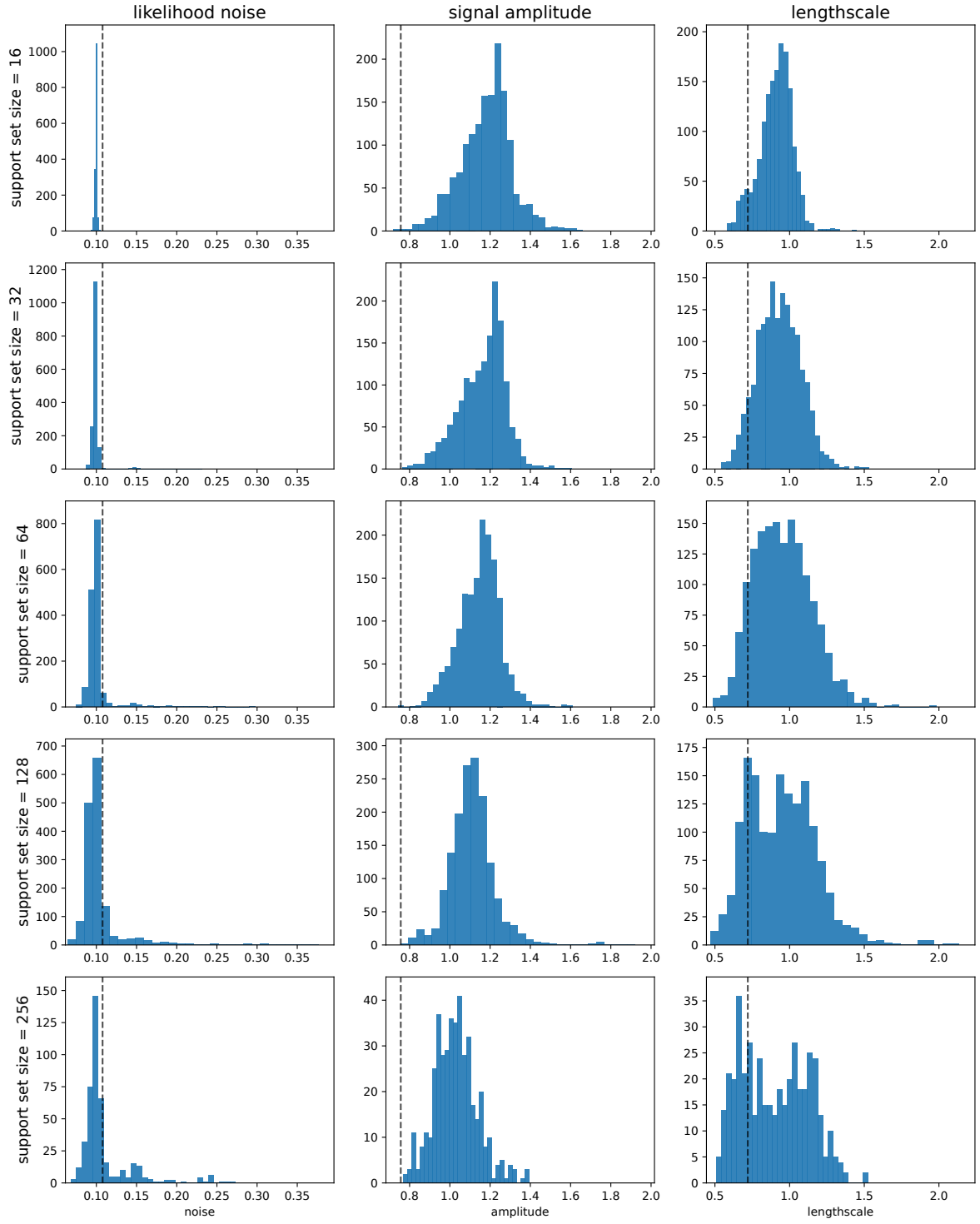


Figure B.1 Visualisation of the distributions of the optimal ADKF-IFT base kernel parameters θ against the optimal DKT base kernel parameters on all FS-Mol test classification tasks. In each plot, the blue histogram represents the empirical distribution of a ADKF-IFT base kernel parameter (x-axis: hyperparameter value, y-axis: frequency), and the black dotted line denotes the value of that base kernel parameter in DKT.

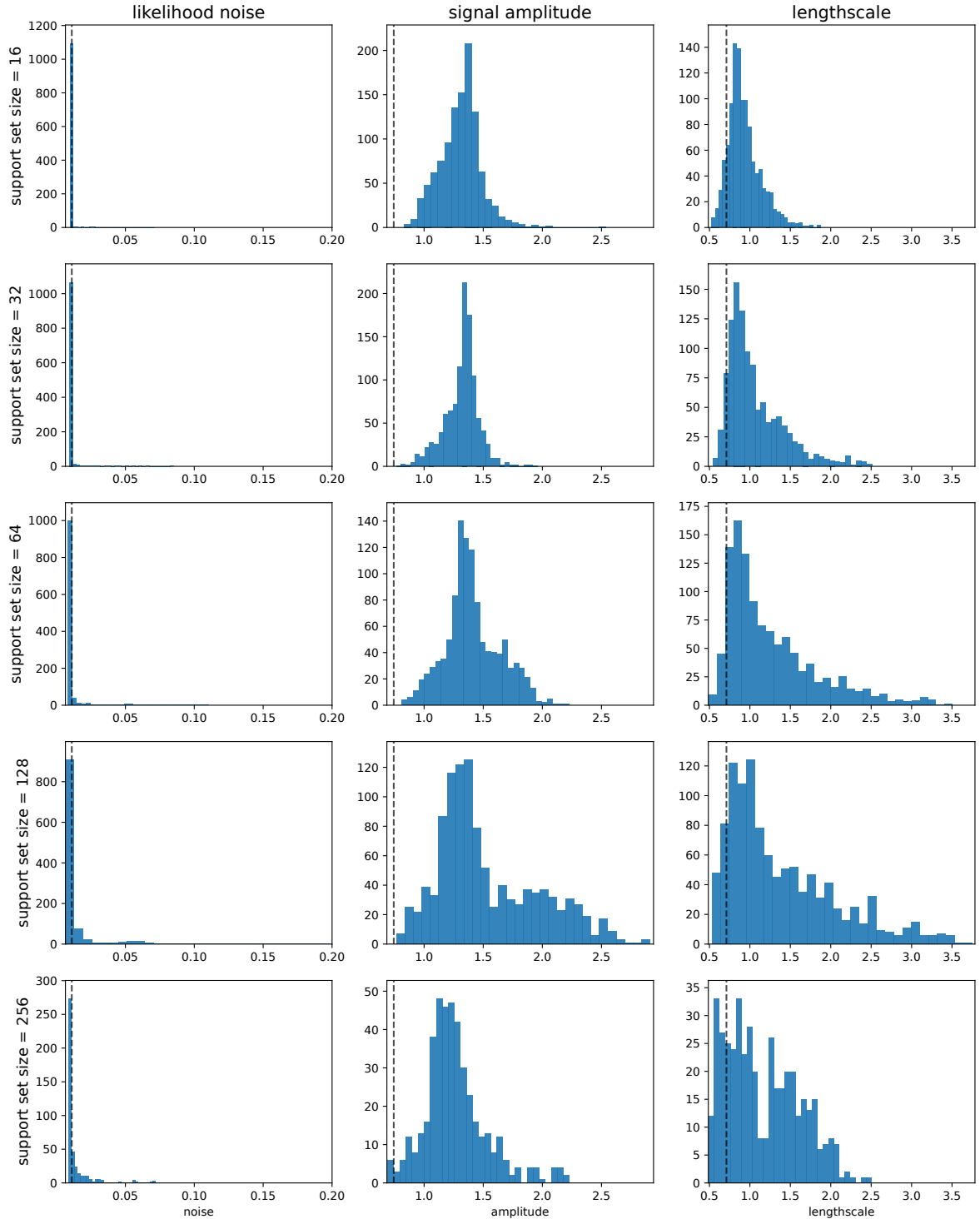


Figure B.2 Visualisation of the distributions of the optimal ADKF-IFT base kernel parameters θ against the optimal DKT base kernel parameters on all FS-Mol test regression tasks. In each plot, the blue histogram represents the empirical distribution of a ADKF-IFT base kernel parameter (x-axis: hyperparameter value, y-axis: frequency), and the black dotted line denotes the value of that base kernel parameter in DKT.

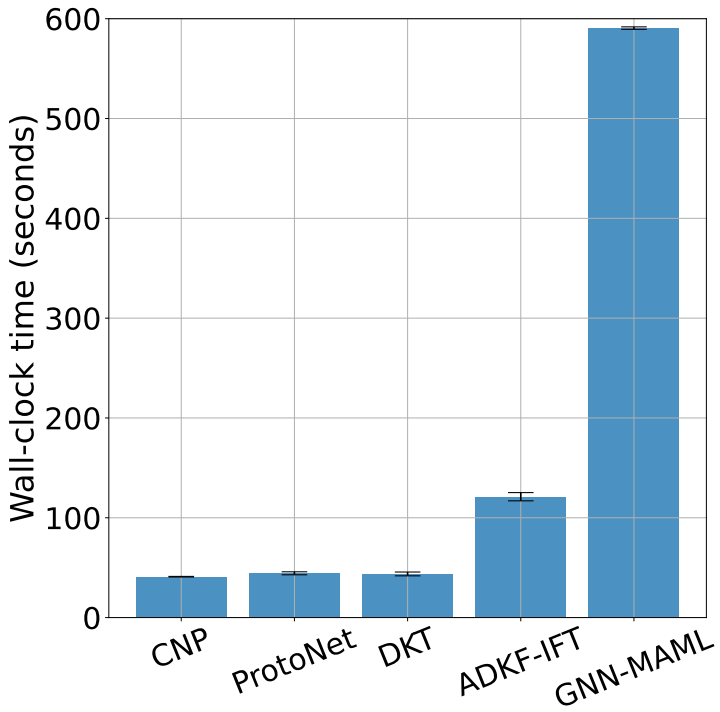


Figure B.3 Wall-clock time consumed (with standard errors) when meta-testing on a pre-defined set of FS-Mol classification tasks using each of the compared meta-learning methods.

B.5 Meta-Testing Cost on FS-Mol

Figure B.3 shows the meta-testing costs of all compared meta-learning methods in terms of wall-clock time[1] on a pre-defined set of FS-Mol classification tasks. These experiments were run on a single NVIDIA GeForce RTX 2080 Ti GPU. It can be seen that ADKF-IFT is $\sim$ 2.5x slower than CNP, ProtoNet, and DKT, but still much faster than GNN-MAML. We did not report the wall-clock time for PAR as it is extremely memory intensive (PAR takes $>$ 10x memory than ADKF-IFT does) and thus cannot be run on a single GPU.

B.6 Reproducibility Statement

Our open-source implementation of ADKF-IFT for reproducing the experimental results in Chapter 3 can be found at https://github.com/Wenlin-Chen/ADKF-IFT, which is based on a fork from the FS-Mol repository (https://github.com/microsoft/FS-Mol) (Stanley et al., 2021) and PAR repository (https://github.com/tata1661/PAR-NeurIPS21) (Wang et al., 2021).

[1]We acknowledge that wall-clock time may not be the best metric for measuring the costs, since some meta-learning methods could be parallelised, which will reduce the wall-clock time accordingly. An alternative metric would be multiply–accumulate operation (MAC). However, it is difficult to obtain the accurate number of MACs due to the opaqueness of the GP modules used.

Appendix C

Supplementary Material for Chapter 4

C.1 Proof of Theorem 4.3.2

Theorem 4.3.2. *Let $\theta \coloneqq (\phi, \{w_\tau\}_{\tau=1}^{N_\tau})$ and $\theta' \coloneqq (\phi', \{w'_\tau\}_{\tau=1}^{N_\tau})$ be any two sets of multi-task regression model parameters such that*

$$p_\theta(y|x,\tau) = p_{\theta'}(y|x,\tau), \quad \forall \tau, x, y. \tag{C.1}$$

Assume that Span(Im(h_ϕ)) $= \mathbb{R}^{d_z}$, i.e., the vectors in the image of the feature extractor h_ϕ span the whole $\mathbb{R}^{d_z}$. Suppose that there exist d_z tasks $\{\tau_i\}_{i=1}^{d_z} \subseteq \{1, \cdots, N_\tau\}$ such that the set of the regression weights $\{w_{\tau_i}\}_{i=1}^{d_z}$ are linearly independent. Then, the data representations of the MTRN are linearly identifiable.

Proof. By the assumption that the predictive likelihoods for the two sets of parameters θ' and θ are equal, we have

$$p_{\theta'}(y|x,\tau) = p_\theta(y|x,\tau), \quad \forall \tau, x, y, \tag{C.2}$$

$$\implies \mathcal{N}(y|\hat{y}_{\phi',w'_\tau}(x), \sigma^2_{r,\tau}) = \mathcal{N}(y|\hat{y}_{\phi,w_\tau}(x), \sigma^2_{r,\tau}), \quad \forall \tau, x, y, \tag{C.3}$$

$$\implies \mathcal{N}(y|h_{\phi'}(x)^\top w'_\tau, \sigma^2_{r,\tau}) = \mathcal{N}(y|h_\phi(x)^\top w_\tau, \sigma^2_{r,\tau}), \quad \forall \tau, x, y, \tag{C.4}$$

$$\implies h_{\phi'}(x)^\top w'_\tau = h_\phi(x)^\top w_\tau, \quad \forall \tau, x, y. \tag{C.5}$$

By the assumption that Span(Im(h_ϕ)) $= \mathbb{R}^{d_z}$, there must exist d_z input data points $x_1, \cdots, x_{d_z}$ such that the matrix $H \coloneqq [h_\phi(x_1), \cdots, h_\phi(x_{d_z})] \in \mathbb{R}^{d_z \times d_z}$ is invertible. By the assumption that there exist d_z tasks $\{\tau_i\}_{i=1}^{d_z}$ such that the set of regression weights $\{w_{\tau_i}\}_{i=1}^{d_z}$ are linearly independent, we construct an invertible matrix $W \coloneqq [w_{\tau_1}, \cdots, w_{\tau_{d_z}}] \in \mathbb{R}^{d_z \times d_z}$. For $h_{\phi'}$, we

similarly define $H' \coloneqq [h_{\phi'}(x_1), \cdots, h_{\phi'}(x_{d_z})] \in \mathbb{R}^{d_z \times d_z}$ and $W' \coloneqq [w'_{\tau_1}, \cdots, w'_{\tau_{d_z}}] \in \mathbb{R}^{d_z \times d_z}$. Note that H' and W' are not necessarily invertible.

Now, we evaluate Equation (C.5) at the d_z input data points $x_1, \cdots, x_{d_z}$ and d_z tasks $\tau_1, \cdots, \tau_{d_z}$ defined above, which gives us the following linear equation:

$$(H')^\top W' = H^\top W. \tag{C.6}$$

Since both H and W are invertible by assumption and the weight matrices W and W' do not depend on the input data x, the matrix W' must be invertible.

Now, evaluating Equation (C.5) at the d_z tasks $\tau_1, \cdots, \tau_{d_z}$, we have

$$(W')^\top h_{\phi'}(x) = W^\top h_\phi(x), \quad \forall x \tag{C.7}$$

$$\implies h_{\phi'}(x) = (W')^{-\top} W^\top h_\phi(x), \quad \forall x \tag{C.8}$$

$$\implies h_{\phi'}(x) = A h_\phi(x), \quad \forall x. \tag{C.9}$$

Note that we have shown that $A \coloneqq (W')^{-\top} W^\top$ is invertible. This completes the proof. □

C.2 Proof of Theorem 4.3.5

Theorem 4.3.5. *Let $u \coloneqq [y, \tau]$ denote the conditioning variable and $k \coloneqq 2d_z$. Assume that the learned and ground-truth linear transformations A and A_* are invertible. Suppose that there exist $k+1$ points $u_0, u_1, \cdots, u_k$ such that the matrix*

$$L \coloneqq [\eta(u_1) - \eta(u_0), \cdots, \eta(u_k) - \eta(u_0)] \tag{C.10}$$

is invertible, where

$$\eta(u) \coloneqq \begin{bmatrix} \Lambda_\tau^{-1} a_\tau \\ -\frac{1}{2} \mathit{diag}(\Lambda_\tau^{-1}) \end{bmatrix} \in \mathbb{R}^k \tag{C.11}$$

are the natural parameters of $p_\zeta(z|u)$. Assume that Equation (4.16) has a unique solution and that the optimisation procedure for Equation (4.16) converges to the optimal marginal likelihood

$$p_*(h|y,\tau) \coloneqq \mathcal{N}(h|\mu_\tau^*, \Sigma_\tau^*) \tag{C.12}$$

under standard regularity conditions for maximum marginal likelihood estimators (Gurland, 1954), i.e.,

$$p_{\psi'}(h|y,\tau) = p_*(h|y,\tau), \quad \forall h, y, \tau, \tag{C.13}$$

where μ_τ^* and Σ_τ^* are defined in Equation (4.15) but with the ground-truth linear transformation A_*, ground-truth causal indicators c_τ^* and ground-truth spurious coefficients γ_τ^*. Then, the latent factors recovered by MTLCM are guaranteed to be point-wise identifiable.

Proof. We first rewrite the density of the conditional prior in the exponential family form:

$$p_\zeta(z|u) = Z(u)^{-1} \exp\left(T(z)^\top \eta(u)\right), \tag{C.14}$$

where the normalising constant is given by

$$Z(u) = (2\pi)^{\frac{d}{2}} |\det(\Lambda_\tau)|^{\frac{1}{2}} \exp\left(\frac{1}{2} a_\tau^\top \Lambda_\tau^{-1} a_\tau\right), \tag{C.15}$$

the sufficient statistics are defined as

$$T(z) = \begin{bmatrix} z \\ z \odot z \end{bmatrix} \in \mathbb{R}^k, \tag{C.16}$$

and the natural parameters are defined as

$$\eta(u) = \begin{bmatrix} \Lambda_\tau^{-1} a_\tau \\ -\frac{1}{2}\text{diag}(\Lambda_\tau^{-1}) \end{bmatrix} \in \mathbb{R}^k. \tag{C.17}$$

We also rewrite the likelihood $p_A(h|z)$ using the noise distribution $p_{\varepsilon_o}(\varepsilon_o) \coloneqq \mathcal{N}(\varepsilon_o|0, \sigma_o^2 I)$:

$$p_A(h|z) = \mathcal{N}(h|Az, \sigma_o^2 I) = \mathcal{N}(h - Az|0, \sigma_o^2 I) = p_{\varepsilon_o}(h - Az). \tag{C.18}$$

Let A_* be the ground-truth transformation matrix such that $z^* = A_*^{-1} h$. Denote the ground-truth task-specific variables associated with each task τ by $\psi_*(\tau) = \{c_\tau^*, \gamma_\tau^*\}$. The proof starts off by using the fact that we have maximised the marginal likelihood as shown in Equation (4.14) with respect to A and ψ for all tasks. This means that the marginal likelihoods of the two models are identical:

$$p_{A,\psi}(h|u) = p_{A_*,\psi_*}(h|u), \quad \forall h, u. \tag{C.19}$$

The goal is to show that the latent factors $z = A^{-1}h$ recovered by our model and the ground-truth latent factor $z^* = A_*^{-1}h$ are identical up to permutations and scaling for all h. Starting

from the equality of the two marginal likelihoods as shown in Equation (C.19), we have

$$p_{A,\psi}(h|u) = p_{A_*,\psi_*}(h|u) \tag{C.20}$$

$$\implies \int p_A(h|z)p_\zeta(z|u)\,\mathrm{d}z = \int p_{A_*}(h|z)p_{\zeta_*}(z|u)\,\mathrm{d}z \tag{C.21}$$

$$\implies \int p_{\varepsilon_o}(h - Az)p_\zeta(z|u)\,\mathrm{d}z = \int p_{\varepsilon_o}(h - A_*z)p_{\zeta_*}(z|u)\,\mathrm{d}z \tag{C.22}$$

$$\implies \int p_{\varepsilon_o}(h - \bar{h})p_\zeta(A^{-1}\bar{h}|u)|\det(A)|^{-1}\,\mathrm{d}\bar{h} = \int p_{\varepsilon_o}(h - \hat{h})p_{\zeta_*}(A_*^{-1}\hat{h}|u)|\det(A_*)|^{-1}\,\mathrm{d}\hat{h} \tag{C.23}$$

$$\implies \int p_{\varepsilon_o}(h - \bar{h})\tilde{p}_{\psi,u}(\bar{h})\,\mathrm{d}\bar{h} = \int p_{\varepsilon_o}(h - \hat{h})\tilde{p}_{\psi_*,u}(\hat{h})\,\mathrm{d}\hat{h} \tag{C.24}$$

$$\implies (p_{\varepsilon_o} * \tilde{p}_{\psi,u})(h) = (p_{\varepsilon_o} * \tilde{p}_{\psi_*,u})(h) \tag{C.25}$$

$$\implies \mathcal{F}[p_{\varepsilon_o}]\mathcal{F}[\tilde{p}_{\psi,u}] = \mathcal{F}[p_{\varepsilon_o}]\mathcal{F}[\tilde{p}_{\psi_*,u}] \tag{C.26}$$

$$\implies \mathcal{F}[\tilde{p}_{\psi,u}] = \mathcal{F}[\tilde{p}_{\psi_*,u}] \tag{C.27}$$

$$\implies \tilde{p}_{\psi,u}(h) = \tilde{p}_{\psi_*,u}(h) \tag{C.28}$$

$$\implies p_\zeta(A^{-1}h|u)|\det(A)|^{-1} = p_{\zeta_*}(A_*^{-1}h|u)|\det(A_*)|^{-1} \tag{C.29}$$

$$\implies T(A^{-1}h)^\top\eta(u) - \log Z(u) - \log|\det(A)| = T(A_*^{-1}h)^\top\eta_*(u) - \log Z_*(u) - \log|\det(A_*)|, \tag{C.30}$$

where

- Equation (C.23) follows by the definitions $\bar{h} \coloneqq Az$ and $\hat{h} \coloneqq A_*z$,
- Equation (C.24) follows by the definition $\tilde{p}_{A,\psi,u}(h) \coloneqq p_\psi(A^{-1}h|u)|\det(A)|^{-1}$,
- $*$ in Equation (C.25) denotes the convolution operator,
- $\mathcal{F}$ in Equation (C.26) denotes the Fourier transform operator,
- Equation (C.27) follows since the characteristic function $\mathcal{F}[p_{\varepsilon_o}]$ of the Gaussian noise ε_o is non-zero almost everywhere.

Now we evaluate Equation (C.30) at $u = u_0, u_1, \cdots, u_k$ from our assumption to obtain $k+1$ such equations and subtract the first equation from the remaining k equations to obtain the following k equations:

$$T(A^{-1}h)^\top(\eta(u_l) - \eta(u_0)) + \log\frac{Z(u_0)}{Z(u_l)} = T(A_*^{-1}h)^\top(\eta_*(u_l) - \eta_*(u_0)) + \log\frac{Z_*(u_0)}{Z_*(u_l)}, \tag{C.31}$$

where $l = 1, \cdots, k$. Putting those k equations in the matrix-vector form gives

$$L^\top T(A^{-1}h) = L_*^\top T(A_*^{-1}h) + b, \tag{C.32}$$

where $b_l \coloneqq \log\frac{Z_*(u_0)Z(u_l)}{Z_*(u_l)Z(u_0)}$, L is the invertible matrix defined in the assumption, and L_* is similarly defined for the second model but is not necessarily invertible. Since L is invertible,

we can left multiply Equation (C.32) by $L^{-\top}$ to obtain

$$T(A^{-1}h) = MT(A_*^{-1}h) + r, \tag{C.33}$$

where $M \coloneqq L^{-\top}L_*^{\top}$ and $r \coloneqq L^{-\top}b$. We note that our assumption only says L is invertible and tells us nothing about L_*. Therefore, we need to show that M is invertible. Let $h_l \coloneqq Az_l$ for $l = 0 \cdots, k$. We evaluate Equation (C.33) at these $k+1$ points to obtain $k+1$ such equations, and subtract the first equation from the remaining k equations. This gives us

$$[T(z_1)-T(z_0), \cdots, T(z_k)-T(z_0)] = M[T(A_*^{-1}h_1)-T(A_*^{-1}h_0), \cdots, T(A_*^{-1}h_k)-T(A_*^{-1}h_0)]. \tag{C.34}$$

We denote Equation (C.34) by $R \coloneqq MR_*$. We need to show that for any given z_0, there exist k points $z_1, \cdots, z_k$ such that the columns of R are linearly independent. Suppose, for contradiction, that the columns of R would never be linearly independent for any $z_1, \cdots, z_k$. Then the function $g(z) \coloneqq T(z) - T(z_0)$ would live in a $k-1$ or lower dimensional subspace, and therefore we would be able to find a non-zero vector $\lambda \in \mathbb{R}^k$ orthogonal to that subspace. This would imply that $(T(z) - T(z_0))^{\top}\lambda = 0$ and thus $T(z)^{\top}\lambda = T(z_0)^{\top}\lambda = \text{const.}$ for all z, which contradicts the fact that our conditionally factorised multivariate Gaussian prior $p_\zeta(z|u)$ is strongly exponential (Khemakhem et al., 2020a). This shows that there must exist k points $z_1, \cdots, z_k$ such that the columns of R are linearly independent for any given z_0. Therefore, R is invertible. Since $R = MR_*$ and M is not a function of z, this tells us that M must be invertible.

Now that we have shown that M is invertible, the next step is to show that M is a block transformation matrix. We define a linear function $l(z) \coloneqq A_*^{-1}Az$. Now, Equation (C.33) becomes

$$T(z) = MT(l(z)) + r. \tag{C.35}$$

We first show that the linear function l is a point-wise function. We differentiate both sides of the above equation with respect to z_s and z_t ($\forall s \neq t$) to obtain:

$$\frac{\partial T(z)}{\partial z_s} = M\sum_{i=1}^{d_z}\frac{\partial T(l(z))}{\partial l_i(z)}\frac{\partial l_i(z)}{\partial z_s}, \tag{C.36}$$

$$\frac{\partial^2 T(z)}{\partial z_s \partial z_t} = M\sum_{i=1}^{d_z}\sum_{j=1}^{d_z}\frac{\partial^2 T(l(z))}{\partial l_i(z)\partial l_j(z)}\frac{\partial l_j(z)}{\partial z_t}\frac{\partial l_i(z)}{\partial z_s} + M\sum_{i=1}^{d_z}\frac{\partial T(l(z))}{\partial l_i(z)}\frac{\partial^2 l_i(z)}{\partial z_s \partial z_t}. \tag{C.37}$$

Since the prior $p_\zeta(z|u)$ is conditionally factorised, the second-order cross derivatives of the sufficient statistics are zeros. Therefore, the second equation above can be simplified as

follows:

$$0 = \frac{\partial^2 T(z)}{\partial z_s \partial z_t} \tag{C.38}$$

$$= M \sum_{i=1}^{d_z} \frac{\partial^2 T(l(z))}{\partial l_i(z)^2} \frac{\partial l_i(z)}{\partial z_t} \frac{\partial l_i(z)}{\partial z_s} + M \sum_{i=1}^{d_z} \frac{\partial T(l(z))}{\partial l_i(z)} \frac{\partial^2 l_i(z)}{\partial z_s \partial z_t} \tag{C.39}$$

$$= MT''(z) l'_{s,z}(z) + MT'(z) l''_{s,z}(z) \tag{C.40}$$

$$= MT'''(z) l'''_{s,z}(z), \tag{C.41}$$

where

$$T''(z) \coloneqq \left[\frac{\partial^2 T(l(z))}{\partial l_1(z)^2}, \cdots, \frac{\partial^2 T(l(z))}{\partial l_{d_z}(z)^2} \right] \in \mathbb{R}^{k \times d_z}, \tag{C.42}$$

$$l'_{s,z}(z) \coloneqq \left[\frac{\partial l_1(z)}{\partial z_t} \frac{\partial l_1(z)}{\partial z_s}, \cdots, \frac{\partial l_{d_z}(z)}{\partial z_t} \frac{\partial l_{d_z}(z)}{\partial z_s} \right]^\top \in \mathbb{R}^{d_z}, \tag{C.43}$$

$$T'(z) \coloneqq \left[\frac{\partial T(l(z))}{\partial l_1(z)}, \cdots, \frac{\partial T(l(z))}{\partial l_{d_z}(z)} \right] \in \mathbb{R}^{k \times d_z}, \tag{C.44}$$

$$l''_{s,z}(z) \coloneqq \left[\frac{\partial^2 l_1(z)}{\partial z_s \partial z_t}, \cdots, \frac{\partial^2 l_{d_z}(z)}{\partial z_s \partial z_t} \right]^\top \in \mathbb{R}^{d_z}, \tag{C.45}$$

$$T'''(z) \coloneqq [T''(z), T'(z)] \in \mathbb{R}^{k \times k}, \tag{C.46}$$

$$l'''_{s,z}(z) = [l'_{s,z}(z)^\top, l''_{s,z}(z)^\top]^\top \in \mathbb{R}^k. \tag{C.47}$$

By Lemma 5 in Khemakhem et al. (2020a) and the fact that $k = 2d_z$, we have that the rank of $T'''(z)$ is $2d_z$ and thus it is invertible for all z. Since M is also invertible, we have that $MT'''(z)$ is invertible. Since $MT'''(z) l'''_{s,z}(z) = 0$, it must be that $l'''_{s,z}(z) = 0,\ \forall z$. In particular, this means that $l'_{s,z}(z) = 0,\ \forall s \neq t$ for all z, which shows that the linear function $l(z) = A_*^{-1} A z$ is a point-wise linear function.

Now, we are ready to show that M is a block transformation matrix. Without loss of generality, we assume that the permutation in the point-wise linear function l is the identity. That is, $l(z) = [l_1 z_1, \cdots, l_{d_z} z_{d_z}]^\top$ for some linear univariate scalars $l_1, \cdots, l_{d_z} \in \mathbb{R}$. Since A and A_* are invertible, we have that $l^{-1}(z) = [l_1^{-1} z_1, \cdots, l_{d_z}^{-1} z_{d_z}]^\top$. Define

$$\bar{T}(l(z)) \coloneqq T(l(z)) + M^{-1} r \tag{C.48}$$

and plug it into Equation (C.35) gives:

$$T(z) = M \bar{T}(l(z)). \tag{C.49}$$

We then apply l^{-1} to z at both sides of the Equation (C.49) to obtain

$$T(l^{-1}(z)) = M\bar{T}(z). \tag{C.50}$$

Since l is a point-wise function, for a given $q \in \{1, \cdots, k\}$, we have that for any s such that $q \neq s$ and $q \neq 2s$

$$0 = \frac{\partial T(l^{-1}(z))_q}{\partial z_s} = \sum_{j=1}^{k} M_{q,j} \frac{\partial \bar{T}(z)_j}{\partial z_s}. \tag{C.51}$$

Since the entries in $\bar{T}(z)$ are linearly independent, it must be that $M_{q,j} = 0$ for any j such that $\frac{\partial \bar{T}(z)_j}{\partial z_s} \neq 0$. This includes the entries j in $\bar{T}(z)$ which depend on z_s (i.e., $j = s$ and $j = 2s$). Note that this holds true for any s such that $q \neq s$ and $q \neq 2s$. Therefore, when q is the index of an entry in the sufficient statistics T that corresponds to z_i (i.e., $q = i$ or $q = 2i$, and $i \neq s$), the only possible non-zero $M_{q,j}$ for j are the ones that map between $T_i(z_i)$ and $\bar{T}_i(l_i(z_i))$, where T_i are the factors in T that depend on z_i and $\bar{T}_i$ are similarly defined. This shows that M is a block transformation matrix for each block $[z_i, z_i^2]$ with scaling factor l_i. That is, the only possible non-zero element in M are $M_{i,i}$, $M_{i,2i}$, $M_{2i,i}$, and $M_{2i,2i}$ for all $i \in \{1, \cdots, d_z\}$.

Furthermore, for any $i \in \{1, \cdots, d_z\}$ we have that

$$l_i^{-1} = \frac{\partial T(l^{-1}(z))_i}{\partial z_i} = \sum_{j=1}^{k} M_{i,j} \frac{\partial \bar{T}(z)_j}{\partial z_i} = M_{i,i} + 2M_{i,2i} z_i, \tag{C.52}$$

$$2l_i^{-1} z_i = \frac{\partial T(l^{-1}(z))_{2i}}{\partial z_i} = \sum_{j=1}^{k} M_{2i,j} \frac{\partial \bar{T}(z)_j}{\partial z_i} = M_{2i,i} + 2M_{2i,2i} z_i. \tag{C.53}$$

This implies that $M_{i,2i} = 0$ and $M_{2i,i} = 0$ for any $i \in \{1, \cdots, d_z\}$, and $M_{i,i} = l_i^{-1}$ for $i \in \{1, \cdots, k\}$, which reduces M from a block transformation matrix to a point-wise permutation and scaling matrix. In particular, this means that the latent factors z_i are identifiable up to point-wise permutations and scaling, with the transformation matrix $P \in \mathbb{R}^{d_z \times d_z}$ defined by the first d_z rows and d_z columns of M:

$$A^{-1}h = PA_*^{-1}h + r \quad \iff \quad h = AP(A_*^{-1}h) + Ar. \tag{C.54}$$

Since h is linearly identifiable by assumption, it must be that $Ar = 0$ by Definition 4.3.1. Since A is invertible by assumption, it must be that $r = 0$. Therefore, we have

$$A^{-1}h = PA_*^{-1}h. \tag{C.55}$$

This completes the proof. □

C.3 Derivation of the Conditionally Factorised Prior

Since no prior knowledge is assumed for the task-specific regression weights $w_\tau \in \mathbb{R}^{d_z}$, we put an improper uniform prior over w_τ for all tasks τ:

$$p_w(w_\tau) \propto 1, \tag{C.56}$$

which can be thought of as a Gaussian prior with infinite variance:

$$p_w(w_\tau) \coloneqq \lim_{\kappa\to\infty} q_\kappa(w_\tau) \coloneqq \lim_{\kappa\to\infty} \mathcal{N}(w_\tau|0,\kappa^2 I). \tag{C.57}$$

Note that improper distributions do not need to integrate to one but may be used as priors (e.g., non-informative priors) in Bayesian inference to produce valid posterior distributions in certain cases (Bishop, 2006; Murphy, 2012).

We marginalise out the weights w_τ from the likelihood $p_\zeta(y|z_c,w_\tau,\tau) = \mathcal{N}(y|(w_\tau \odot c_\tau)^\top z, \sigma_p^2)$ over the improper uniform prior $p_w(w_\tau)$, which makes the marginal distribution $p_\zeta(y|z_c,\tau)$ an improper uniform distribution over the target variable y:

$$\begin{aligned}
p_\zeta(y|z_c,\tau) &= \int p_\zeta(y|z_c,w_\tau,\tau)p_w(w_\tau)\,\mathrm{d}w_\tau && \text{(C.58)}\\
&= \int p_\zeta(y|z_c,w_\tau,\tau)\lim_{\kappa\to\infty} q_\kappa(w_\tau)\,\mathrm{d}w_\tau && \text{(C.59)}\\
&= \lim_{\kappa\to\infty}\int p_\zeta(y|z_c,w_\tau,\tau)q_\kappa(w_\tau)\,\mathrm{d}w_\tau && \text{(C.60)}\\
&= \lim_{\kappa\to\infty}\mathcal{N}(y|0,\kappa^2(c_\tau\odot z)^\top(c_\tau\odot z)+\sigma_p^2) && \text{(C.61)}\\
&\propto 1. && \text{(C.62)}
\end{aligned}$$

Since improper distributions do not need to integrate to one, we define the density of the above marginal distribution to be a constant C everywhere: $p_\zeta(y|z_c,\tau) \equiv C$. Then, this results in a valid posterior distribution according to the Bayes' rule:

$$\begin{aligned}
p_\zeta(z|y,\tau) &= \frac{p_\zeta(z_c|\tau)p_\zeta(y|z_c,\tau)p_\zeta(z_s|y,\tau)}{\iint p_\zeta(z_c|\tau)p_\zeta(y|z_c,\tau)p_\zeta(z_s|y,\tau)\,\mathrm{d}z_s\,\mathrm{d}z_c} && \text{(C.63)}\\
&= \frac{p_\zeta(z_c|\tau)p_\zeta(z_s|y,\tau)}{\iint p_\zeta(z_c|\tau)p_\zeta(z_s|y,\tau)\,\mathrm{d}z_s\,\mathrm{d}z_c} && \text{(C.64)}\\
&= p_\zeta(z_c|\tau)p_\zeta(z_s|y,\tau). && \text{(C.65)}
\end{aligned}$$

Since $p_\zeta(z_c|\tau)$ factorizes over the causal latent factors and $p_\zeta(z_s|y,\tau)$ factorizes over the spurious latent factors, we conclude that the structured conditional prior $p_\zeta(z|y,\tau)$ factorises over all latent factors z.

Furthermore, we verify that the compact expressions for the mean a_τ and variance Λ_τ of $p_\zeta(z|y,\tau)$ in Equation (4.12) are correct. Recall that Equation (4.7) tells us that

$$p_\zeta(z_c|\tau) = \mathcal{N}(z_c|0, I), \tag{C.66}$$

and Equation (4.9) tells us that

$$p_\zeta(z_s|y,\tau) = \mathcal{N}(z_s|y\gamma_\tau, \sigma_s^2 I). \tag{C.67}$$

Recall that the compact expressions given by Equation (4.12) are

$$\begin{aligned} a_\tau &\coloneqq y\gamma_\tau \circ (1 - c_\tau), \\ \Lambda_\tau &\coloneqq \text{diag}(\sigma_s^2(1 - c_\tau) + c_\tau). \end{aligned} \tag{C.68}$$

For any causal latent variable z_i, we have $c_{\tau,i} = 1$ and therefore $a_{\tau,i} = 0$ and $\Lambda_{\tau,i} = 1$. For any spurious latent variable z_j, we have $c_{\tau,j} = 0$ and therefore $a_{\tau,j} = y\gamma_{\tau,j}$ and $\Lambda_{\tau,j} = \sigma_s^2$. This verifies that Equation (4.12) is correct.

C.4 Derivation of the Marginal Likelihood for MTLCM

The marginal likelihood for MTRN given by Equation (4.14) is

$$\begin{aligned} p_\psi(h|y,\tau) &= \int p_A(h|z) p_\zeta(z|y,\tau)\, \mathrm{d}z \\ &\coloneqq \mathcal{N}(h|\mu_\tau, \Sigma_\tau), \end{aligned} \tag{C.69}$$

where $p_A(h|z) = \mathcal{N}(h|Az, \sigma_o^2 I)$ and $p_\zeta(z|y,\tau) = \mathcal{N}(z|a_\tau, \Lambda_\tau)$. Equivalently, we can rewrite the likelihood in the following form:

$$h = Az + \varepsilon, \tag{C.70}$$

where $p_\varepsilon(\varepsilon) = \mathcal{N}(\varepsilon|0, \sigma_o^2 I)$. Since both $p_A(h|z)$ and $p_\zeta(z|y,\tau)$ are linear Gaussians, we can derive closed-form expression for the mean μ_τ and covariance Σ_τ using moment matching:

$$\mu_\tau = \mathbb{E}_{p_\zeta(z|y,\tau)}[h] \tag{C.71}$$

$$= A\mathbb{E}_{p_\zeta(z|y,\tau)}[z] \tag{C.72}$$

$$= Aa_\tau \tag{C.73}$$

$$= yA(\gamma_\tau \odot (1 - c_\tau)), \tag{C.74}$$

and

$$\Sigma_\tau = \text{Var}_{p_\zeta(z|y,\tau)}[h] \tag{C.75}$$
$$= A\text{Var}_{p_\zeta(z|y,\tau)}[z]A^\top + \text{Var}_{p_\varepsilon(\varepsilon)}[\varepsilon] \tag{C.76}$$
$$= A\Lambda_\tau A^\top + \sigma_o^2 I \tag{C.77}$$
$$= A\text{diag}(\sigma_s^2(1-c_\tau)+c_\tau)A^\top + \sigma_o^2 I. \tag{C.78}$$

This verifies that Equation (4.14) is correct.

C.5 Model Configurations

In Stage 1, the learnable parameters of a multi-task regression network (MTRN) are the feature extractor parameters ϕ and the task-specific regression weights w_τ for all tasks τ. These model parameters are learned by maximum likelihood as defined in Equation (4.4).

In Stage 2, the learnable parameters of a multi-task linear causal model (MTLCM) are the linear transformation A, the causal indicators c_τ for all tasks τ, and the spurious coefficients γ_τ for all tasks τ. These are free parameters learned by maximum marginal likelihood as defined in Equation (4.16). The binary causal indicators c_τ are parameterised as continuous parameters in $\mathbb{R}$ squashed to $[0,1]$ by the sigmoid function. To allow for gradient update of c_τ, we do not binarise the output of the sigmoid function during training; instead, we use a soft version $\tilde{c}_\tau \in [0,1]^{d_z}$ during training. In practice, we find that this works well and all learned values for $\tilde{c}_{\tau,1}$ are very close to either 0 or 1. In the synthetic data setting, we found that the learned causal indicators matched the ground-truth values. In practice, we fix the spurious noise variance σ_s and the observational noise variance σ_0 to some constants using cross validation.

For a fair comparison, we also consider the multi-task extensions of iVAE and iCaRL, MT-iVAE and MT-iCaRL, which include the task variable τ in the conditioning variables u in their conditional priors $p_\zeta(z|u)$, with the task-specific parameter $\psi_\tau = \{v_\tau\}$ is a free parameter to be learned from data, which is the counterpart to $\psi_\tau = \{c_\tau, \gamma_\tau\}$ in our MTLCM but has no explicit interpretations with respect to a causal graph. We set $\dim(v_\tau) = \dim(c_\tau) + \dim(\gamma_\tau)$ to ensure that the baselines have the same degree of flexibility as our MTLCM. The task-specific parameters v_τ are free parameters learned jointly with other parameters in these models by optimising their variational/score matching objectives.

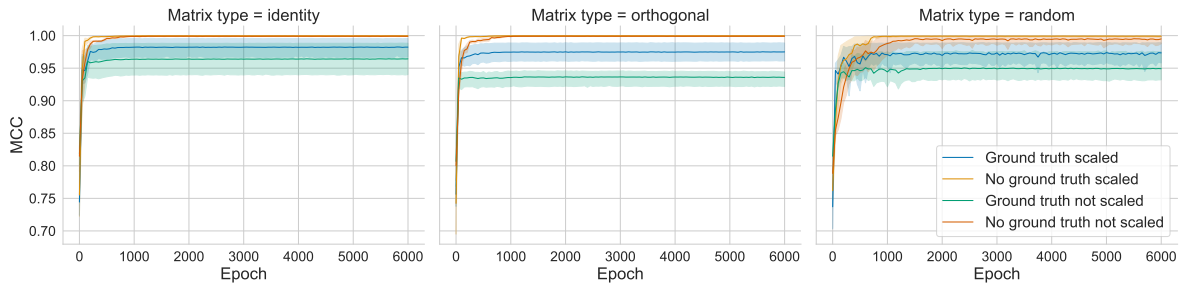


Figure C.1 Convergence of the model in the case of transformations of the latent factors for identity, orthogonal and arbitrary linear transformations. Scaled means standardising the features.

C.6 Ablation Study for the Linear Synthetic Experiment

In Figure C.1, we contrast the effect of training only the linear transformation matrix A in our MTLCM when the ground-truth task variables c_τ^*, γ_τ^* are known to the model, with the more general setting of learning all parameters jointly via maximum marginal likelihood. We assess the convergence of our multi-task linear causal model (MTLCM) across 5 random seeds for increasingly complex linear transformations (i.e., identity, orthogonal, random) for data consisting of 10 latent factors with 2 causal features. Rather than inhibiting convergence, we find that training all parameters jointly leads to improved performance, possibly due to additional flexibility in the parametrisations of the model. For all types of linear transformations, our model succeeds in recovering the ground-truth latent factors. In additional, we find that standardising the features accelerates convergence.

C.7 Reproducibility Statement

Our open-source implementation of MTRN and MTLCM for reproducing the experimental results in Chapter 4 can be found at https://github.com/jdhorwood/mtlcm.

Appendix D

Supplementary Material for Chapter 5

D.1 DiKL Divergence is a Lower Bound of KL Divergence

Below, we show that DiKL divergence is a lower bound of KL divergence for a single kernel $k(x_t|x)$; the extension to multiple kernels is straightforward.

First, we show that the KL divergence between two joint densities $p(x, y) = p(y|x)p(x)$ and $q(x, y) = q(y|x)q(x)$ can be factorized as follows:

$$\begin{aligned}
\mathrm{KL}(p(x,y)||q(x,y)) &= \mathrm{KL}(p(x)p(y|x)||q(x)q(y|x)) && \text{(D.1)}\\
&= \iint p(x)p(y|x)\log\frac{p(x)p(y|x)}{q(x)q(y|x)}\,\mathrm{d}x\,\mathrm{d}y && \text{(D.2)}\\
&= \int p(x)\log\frac{p(x)}{q(x)}\,\mathrm{d}x + \iint p(x)p(y|x)\log\frac{p(y|x)}{q(y|x)}\,\mathrm{d}y\,\mathrm{d}x && \text{(D.3)}\\
&= \mathrm{KL}(p(x)||q(x)) + \mathbb{E}_{p(x)}[\mathrm{KL}(p(y|x)||q(y|x))]. && \text{(D.4)}
\end{aligned}$$

Using the above identity, we have

$$\begin{aligned}
\mathrm{KL}(p(x)||q(x)) &= \mathrm{KL}(p(x)||q(x)) + \mathbb{E}_{p(x)}[\mathrm{KL}(k(x_t|x)||k(x_t|x))] && \text{(D.5)}\\
&= \mathrm{KL}(p(x)k(x_t|x)||q(x)k(x_t|x)) && \text{(D.6)}\\
&= \mathrm{KL}(p(x_t)p(x|x_t)||q(x_t)q(x|x_t)) && \text{(D.7)}\\
&= \mathrm{KL}(p(x_t)||q(x_t)) + \mathbb{E}_{p(x_t)}[\mathrm{KL}(p(x|x_t)||q(x|x_t))] && \text{(D.8)}\\
&\geq \mathrm{KL}(p(x_t)||q(x_t)) && \text{(D.9)}\\
&\coloneqq \mathrm{DiKL}_{k_t}(p(x)||q(x)), && \text{(D.10)}
\end{aligned}$$

where the equality attains when $p = q$, the marginal noisy densities are defined as

$$p(x_t) = \int k(x_t|x)p(x)\,\mathrm{d}x, \tag{D.11}$$

$$q(x_t) = \int k(x_t|x)q(x)\,\mathrm{d}x, \tag{D.12}$$

and the denoising posterior densities are defined as

$$p(x|x_t) = \frac{k(x_t|x)p(x)}{p(x_t)}, \tag{D.13}$$

$$q(x|x_t) = \frac{k(x_t|x)q(x)}{q(x_t)}. \tag{D.14}$$

D.2 Derivation of the Analytical Gradient for R-DiKL

Recall that the gradient of R-DiKL with respect to the model parameter θ is given by

$$\nabla_\theta \mathrm{DiKL}_{k_t}(p_\theta||p_d) = \int p_\theta(x_t)\left(\nabla_{x_t}\log p_\theta(x_t) - \nabla_{x_t}\log p_d(x_t)\right)\frac{\partial x_t}{\partial \theta}\,\mathrm{d}x_t. \tag{D.15}$$

Proof. The R-DiKL at time t is defined as

$$\mathrm{DiKL}_{k_t}(p_\theta||p_d) = \int \left(\log p_\theta(x_t) - \log p_d(x_t)\right)p_\theta(x_t)\,\mathrm{d}x_t. \tag{D.16}$$

We first reparameterise x_t as a function of z and ε:

$$x_t = \alpha_t f_\theta(z) + \sigma_t \varepsilon_t \coloneqq h_\theta(z, \varepsilon_t), \tag{D.17}$$

where $z \sim p_z(z) \coloneqq \mathcal{N}(z|0, I)$ and $\varepsilon_t \sim p_\varepsilon(\varepsilon_t) \coloneqq \mathcal{N}(\varepsilon_t|0, I)$. It then follows that

$$\nabla_\theta \mathrm{DiKL}_{k_t}(p_\theta||p_d) \tag{D.18}$$

$$= \nabla_\theta \int \left(\log p_\theta(x_t) - \log p_d(x_t)\right)p_\theta(x_t)\,\mathrm{d}x_t \tag{D.19}$$

$$= \nabla_\theta \iiint \left(\log p_\theta(x_t) - \log p_d(x_t)\right)\delta(x_t - h_\theta(z,\varepsilon_t))p_z(z)p_\varepsilon(\varepsilon_t)\,\mathrm{d}x_t\,\mathrm{d}z\,\mathrm{d}\varepsilon \tag{D.20}$$

$$= \nabla_\theta \iint \left(\log p_\theta(x_t) - \log p_d(x_t)\right)|_{x_t = h_\theta(z,\varepsilon_t)}p_z(z)p_\varepsilon(\varepsilon_t)\,\mathrm{d}z\,\mathrm{d}\varepsilon \tag{D.21}$$

$$= \iint \left(\nabla_\theta \log p_\theta(x_t) + \nabla_{x_t}\log p_\theta(x_t)\frac{\partial x_t}{\partial \theta} - \nabla_{x_t}\log p_d(x_t)\frac{\partial x_t}{\partial \theta}\right)\Bigg|_{x_t = h_\theta(z,\varepsilon_t)} p_z(z)p_\varepsilon(\varepsilon_t)\,\mathrm{d}z\,\mathrm{d}\varepsilon \tag{D.22}$$

$$= \int \left(\nabla_\theta \log p_\theta(x_t) + \nabla_{x_t}\log p_\theta(x_t)\frac{\partial x_t}{\partial \theta} - \nabla_{x_t}\log p_d(x_t)\frac{\partial x_t}{\partial \theta}\right)p_\theta(x_t)\,\mathrm{d}x_t \tag{D.23}$$

$$= \int \left(\nabla_{x_t}\log p_\theta(x_t)\frac{\partial x_t}{\partial \theta} - \nabla_{x_t}\log p_d(x_t)\frac{\partial x_t}{\partial \theta}\right)p_\theta(x_t)\,\mathrm{d}x_t, \tag{D.24}$$

where the last line follows since

$$\int \nabla_\theta \log p_\theta(x_t) p_\theta(x_t)\,\mathrm{d}x_t = \int \nabla_\theta p_\theta(x_t)\,\mathrm{d}x_t \tag{D.25}$$

$$= \nabla_\theta \int p_\theta(x_t)\,\mathrm{d}x_t \tag{D.26}$$

$$= \nabla_\theta 1 = 0. \tag{D.27}$$

This completes the proof. □

D.3 Derivations of Score Identities

D.3.1 Derivation of Target Score Identity

Proposition 5.3.3 (Target Score Identity). *For any translation-invariant convolution kernel $k(x_t|x) = k(x_t - \alpha_t x)$, we have*

$$\nabla_{x_t} \log p_d(x_t) = \frac{1}{\alpha_t} \int \nabla_x \log p_d(x) p_d(x|x_t)\,\mathrm{d}x, \tag{D.28}$$

where $p_d(x|x_t) \propto k(x_t|x)p_d(x)$ is the denoising posterior distribution.

Proof. Since $k(x_t|x) = k(x_t - \alpha_t x)$ is translation-invariant, we have

$$\nabla_{x_t} \log k(x_t|x) = -\alpha^{-1} \nabla_x \log k(x_t|x). \tag{D.29}$$

But by Bayes rule, we have

$$\nabla_x \log k(x_t|x) = \nabla_x \log p_d(x|x_t) - \nabla_x \log p_d(x). \tag{D.30}$$

Using DSI from Proposition 2.3.1, it then follows that

$$\nabla_{x_t} \log p_d(x_t) = \int \nabla_{x_t} k(x_t|x) p_d(x|x_t)\,\mathrm{d}x \tag{D.31}$$

$$= -\alpha^{-1} \int \nabla_x k(x_t|x) p_d(x|x_t)\,\mathrm{d}x \tag{D.32}$$

$$= \alpha^{-1} \int \left(\nabla_x \log p_d(x) - \nabla_x \log p_d(x|x_t)\right) p_d(x|x_t)\,\mathrm{d}x \tag{D.33}$$

$$= \alpha^{-1} \int \nabla_x \log p_d(x) p_d(x|x_t)\,\mathrm{d}x, \tag{D.34}$$

where the last equality follows since

$$\int \nabla_x \log p_d(x|x_t) p_d(x|x_t)\,\mathrm{d}x = \int \nabla_x p_d(x|x_t)\,\mathrm{d}x \tag{D.35}$$

$$= \nabla_x \int p_d(x|x_t)\,\mathrm{d}x \tag{D.36}$$

$$= \nabla_x 1 = 0. \tag{D.37}$$

This completes the proof. □

D.3.2 Derivation of Mixed Score Identity

Proposition 5.3.4 (Mixed Score Identity). *Using a Gaussian convolution kernel $k(x_t|x) = \mathcal{N}(x_t|\alpha_t x, \sigma_t^2 I)$ with a variance-preserving (VP) scheme $\sigma_t^2 = 1 - \alpha_t^2$, and a convex combination of TSI and DSI with coefficients α_t^2 and $1 - \alpha_t^2$, respectively, we have*

$$\nabla_{x_t} \log p_d(x_t) = \int (\alpha_t(x + \nabla_x \log p_d(x)) - x_t) p_d(x|x_t)\,\mathrm{d}x. \tag{D.38}$$

Proof. For a Gaussian convolution kernel $k(x_t|x) = \mathcal{N}(x_t|\alpha_t x, \sigma_t^2 I)$, DSI becomes

$$\nabla_{x_t} \log p_d(x_t) = \int \nabla_{x_t} \log k(x_t|x) p_d(x|x_t)\,\mathrm{d}x \tag{D.39}$$

$$= \int \nabla_{x_t} \left(-\frac{\|x_t - \alpha x\|^2}{2\sigma_t^2} \right) p_d(x|x_t)\,\mathrm{d}x \tag{D.40}$$

$$= \int \left(\frac{\alpha x - x_t}{\sigma_t^2} \right) p_d(x|x_t)\,\mathrm{d}x. \tag{D.41}$$

Since $\sigma_t^2 = 1 - \alpha_t^2$, it then follows that

$$\nabla_{x_t} \log p_d(x_t) = \int \left(\alpha_t^2 \frac{\nabla_x \log p_d(x)}{\alpha_t} + (1 - \alpha_t^2) \frac{\alpha_t x - x_t}{\sigma_t^2} \right) p_d(x|x_t)\,\mathrm{d}x \tag{D.42}$$

$$= \int (\alpha_t(x + \nabla_x \log p_d(x)) - x_t) p_d(x|x_t)\,\mathrm{d}x. \tag{D.43}$$

This completes the proof. □

D.4 Derivations Regarding Invariance and Equivariance

D.4.1 Proof of Proposition 5.5.1

Proposition 5.5.1. *Let the neural sampler $f_\theta : \mathcal{Z} \to \mathcal{X}$ be an G-equivariant mapping. If the distribution $p_z(Z)$ over the latent space $\mathcal{Z}$ is G-invariant, then the model density $p_\theta(X) = \int \delta(X - f_\theta(Z)) p_z(Z)\,\mathrm{d}Z$ is also G-invariant.*

Proof. For any transformation $g \in G = \mathrm{E}(d) \times \mathbb{S}_n$, we have

$$p_\theta(g \circ X) = \int \delta(g \circ X - f_\theta(Z)) p_z(Z) \, \mathrm{d}Z \tag{D.44}$$

$$= \int \delta(X - f_\theta(g^{-1} \circ Z)) p_z(g^{-1} \circ Z) \, \mathrm{d}Z \tag{D.45}$$

$$= \int \delta(X - f_\theta(Z')) p_z(Z') \, \mathrm{d}Z \tag{D.46}$$

$$= \int \delta(X - f_\theta(Z')) p_z(Z') \, \mathrm{d}Z' \tag{D.47}$$

$$= p_\theta(X), \tag{D.48}$$

where $Z' \coloneqq g^{-1} \circ Z$, and the penultimate line follows since the transformations $g \in G = \mathrm{E}(d) \times \mathbb{S}_n$ preserves the volume. This completes the proof. □

D.4.2 Monte Carlo Score Estimators are G-Equivariant

Section 5.5.2 mentioned that we need a G-equivariant estimator for MSI:

$$\nabla_{X_t} \log p_d(X_t) = \int (\alpha_t(X + \nabla_X \log p_d(X)) - X_t) p_d(X|X_t) \, \mathrm{d}X, \tag{D.49}$$

and that this can be achieved by a broad class of Monte Carlo estimators under mild conditions, including importance sampling and AIS estimators, with different choices of samplers for $p_d(X|X_t)$, such as MALA and HMC. We now provide a detailed discussion below.

Recall that we embed the product group $G = \mathrm{E}(d) \times \mathbb{S}_n$ into an orthogonal group in the nd-dimensional space: $\mathrm{E}(d) \times \mathbb{S}_n \hookrightarrow \mathrm{O}(nd)$ by constraining both $\mathcal{X}$ and $\mathcal{Z}$ to be in the subspace of $\mathbb{R}^{n \times d}$ with zero centre-of-mass. Therefore, both X and X_t are zero-centred, i.e., $X^\top 1 = X_t^\top 1 = 0$. Additionally, we can show that the score $\nabla_X \log p_d(X) = -\nabla_X E(X)$ is also zero-centred.

Proposition D.4.1. *Let $X \in \mathbb{R}^{n \times d}$ be a random variable representing a d-dimensional n-body system. Let $E : \mathbb{R}^{n \times d} \to \mathbb{R}$ be a translation invariant energy function. Then, the gradient of E is translation invariant and zero-centred.*

Proof. Let $t \in \mathbb{R}^d$ be a translation. Given the fact that E is translation invariant:

$$E(X) = E(X + 1t^\top), \tag{D.50}$$

taking gradient with respect to X on both sides, we have

$$\nabla_X E(X) = \nabla_X E(X + 1t^\top). \tag{D.51}$$

Let $X' = X + 1t^\top$. By chain rule, we have

$$\nabla_X E(X) = \nabla_X E(X') = \nabla_{X'} E(X') \nabla_X (X + 1t^\top) = \nabla_{X'} E(X'), \tag{D.52}$$

which shows that the gradient of a translation invariant energy function is also translation invariant.

Now, we show that the gradient of E is always zero-centred. Applying the first-order Taylor expansion at $X = X_0$ to both sides of $E(X) = E(X + 1t^\top)$, we have

$$E(X_0) + \text{trace}\left(\nabla E(X_0)(X - X_0)^\top\right) \tag{D.53}$$
$$= E(X_0 + 1t^\top) + \text{trace}\left(\nabla E(X_0 + 1t^\top)(X - X_0 - 1t^\top)^\top\right) \tag{D.54}$$
$$= E(X_0) + \text{trace}\left(\nabla E(X_0)(X - X_0 - 1t^\top)^\top\right). \tag{D.55}$$

It then follows that

$$0 = \text{trace}\left(\nabla E(X_0) t 1^\top\right) = 1^\top \nabla E(X_0)^\top t. \tag{D.56}$$

Note that this holds for any translation t and any point X_0. Hence, we must have

$$\nabla E(X)^\top 1 = 0, \quad \forall X. \tag{D.57}$$

This completes the proof. □

Therefore, the entire MSI as shown in Equation (D.49) is zero-centred and G-equivariant with respect to X_t. Below, we show that various Monte Carlo estimators for MSI used in Chapter 5 meet these requirements.

For simplicity of notation, we will omit the subscript of the gradient operator ∇ unless there is disambiguity. We will use $\mathcal{N}(X|Y, v)$ to denote isotropic Gaussian distributions for matrices:

$$\mathcal{N}(X|Y, v) \coloneqq \mathcal{N}(\text{vec}(X)|\text{vec}(Y), vI). \tag{D.58}$$

We will use $\bar{\mathcal{N}}$ to denote Gaussian distributions in the zero-centred subspace:

$$\bar{\mathcal{N}}(X|\cdot) \propto \begin{cases} \mathcal{N}(X|\cdot), & \text{if } X^\top 1 = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{D.59}$$

Importance Sampling

We begin with the simplest case, where we estimate Equation (D.49) using importance sampling (IS) with a zero-centred Gaussian proposal as in Akhound-Sadegh et al. (2024):

$$q(X|X_t) \coloneqq \bar{\mathcal{N}}(X|X_t/\alpha_t, \sigma_t^2/\alpha_t^2). \tag{D.60}$$

Formally, the IS estimator works as follows:

$$\int (\alpha_t(X + \nabla \log p_d(X)) - X_t) p_d(X|X_t)\,\mathrm{d}X \tag{D.61}$$

$$= \alpha_t \frac{\int (X + \nabla \log p_d(X)) p_d(X) \bar{\mathcal{N}}(X_t|\alpha_t X, \sigma_t^2)\,\mathrm{d}X}{\int p_d(X) \bar{\mathcal{N}}(X_t|\alpha_t X, \sigma_t^2)\,\mathrm{d}X} - X_t \tag{D.62}$$

$$= \alpha_t \frac{\int (X + \nabla \log p_d(X)) \tilde{p}_d(X) \bar{\mathcal{N}}(X|X_t/\alpha_t, \sigma_t^2/\alpha_t^2)\,\mathrm{d}X}{\int \tilde{p}_d(X) \bar{\mathcal{N}}(X|X_t/\alpha_t, \sigma_t^2/\alpha_t^2)\,\mathrm{d}X} - X_t \tag{D.63}$$

$$\approx \alpha_t \frac{\sum_{l=1}^{L} (X^{(l)} + \nabla \log p_d(X^{(l)})) \tilde{p}_d(X^{(l)})}{\sum_{l'=1}^{L} \tilde{p}_d(X^{(l')})} - X_t, \tag{D.64}$$

where $X^{(1:L)} \sim q(X|X_t) = \bar{\mathcal{N}}(X|X_t/\alpha_t, \sigma_t^2/\alpha_t^2)$, $\tilde{p}_d(X)$ denotes the unnormalised density of the target distribution $p_d(X)$, and the penultimate line follows since

$$q(X|X_t) \propto \bar{\mathcal{N}}(X_t|\alpha_t X, \sigma_t^2) \tag{D.65}$$

and the normalising constant Z from the numerator and denominator are cancelled out. In other words, we draw samples from the proposal $q(X|X_t)$ to target at

$$p_d(X|X_t) \propto p_d(X) \bar{\mathcal{N}}(X_t|\alpha_t X, \sigma_t^2). \tag{D.66}$$

Hence, the importance weight is given by

$$w_{\text{IS}}(X) = \frac{p_d(X) \bar{\mathcal{N}}(X_t|\alpha_t X, \sigma_t^2)}{\bar{\mathcal{N}}(X|X_t/\alpha_t, \sigma_t^2/\alpha_t^2)} \propto p_d(X) \propto \tilde{p}_d(X). \tag{D.67}$$

It is easy to check that this estimator is zero-centred. We now show that this estimator is G-equivariant to X_t. Our derivation closely follows the arguments in Akhound-Sadegh et al. (2024).

Suppose that we apply a transformation $g \in G$ to X_t. Then, we have

$$g \circ X^{(1:L)} \sim \bar{\mathcal{N}}(g \circ X_t/\alpha_t, \sigma_t^2/\alpha_t^2). \tag{D.68}$$

Since the target density $p_d(X)$ (and $\tilde{p}_d(X)$) is G-invariant and its score $\nabla \log p_d(X)$ is G-equivariant, the IS estimator becomes

$$\alpha_t \frac{\sum_{l=1}^{L}(g \circ X^{(l)} + \nabla \log p_d(g \circ X^{(l)}))\tilde{p}_d(g \circ X^{(l)})}{\sum_{l'=1}^{L} \tilde{p}_d(g \circ X^{(l')})} - g \circ X_t \tag{D.69}$$

$$= \alpha_t \frac{\sum_{l=1}^{L}(g \circ X^{(l)} + g \circ \nabla \log p_d(X^{(l)}))\tilde{p}_d(X^{(l)})}{\sum_{l'=1}^{L} \tilde{p}_d(X^{(l')})} - g \circ X_t \tag{D.70}$$

$$= g \circ \left(\alpha_t \frac{\sum_{l=1}^{L}(X^{(l)} + \nabla \log p_d(X^{(l)}))\tilde{p}_d(X^{(l)})}{\sum_{l'=1}^{L} \tilde{p}_d(X^{(l')})} - X_t \right), \tag{D.71}$$

where $X^{(1:L)} \sim \bar{\mathcal{N}}(X_t/\alpha_t, \sigma_t^2/\alpha_t^2)$ which is equivalent to $g \circ X^{(1:L)} \sim \bar{\mathcal{N}}(g \circ X_t/\alpha_t, \sigma_t^2/\alpha_t^2)$. This shows that the IS estimator is G-equivariant to X_t.

Sampling Importance Resampling

Instead of estimating Equation (D.49) by IS, we can also perform sampling importance resampling (SIR) using the IS weight $w_{\text{IS}}(X) \propto \tilde{p}_d(X)$. Specifically, we can draw one sample X^* from the categorical distribution according to the IS weights, where

$$\begin{aligned} X^* &= X^{(l_*)} \\ l_* &\sim \text{Categorical}\left(\frac{\tilde{p}_d(X^{(1)})}{\sum_{l=1}^{L} \tilde{p}_d(X^{(l)})}, \cdots, \frac{\tilde{p}_d(X^{(L)})}{\sum_{l=1}^{L} \tilde{p}_d(X^{(l)})} \right), \\ X^{(1:L)} &\sim \bar{\mathcal{N}}(X_t/\alpha_t, \sigma_t^2/\alpha_t^2). \end{aligned} \tag{D.72}$$

Suppose that we apply $g \in G$ to X_t. Now, SIR becomes

$$\begin{aligned} X^{*\prime} &= g \circ X^{(l_*)} \\ l_* &\sim \text{Categorical}\left(\frac{\tilde{p}_d(g \circ X^{(1)})}{\sum_{l=1}^{L} \tilde{p}_d(g \circ X^{(l)})}, \cdots, \frac{\tilde{p}_d(g \circ X^{(L)})}{\sum_{l=1}^{L} \tilde{p}_d(g \circ X^{(l)})} \right), \\ g \circ X^{(1:L)} &\sim \bar{\mathcal{N}}(g \circ X_t/\alpha_t, \sigma_t^2/\alpha_t^2). \end{aligned} \tag{D.73}$$

Since $X^{(1:L)} \sim \bar{\mathcal{N}}(X_t/\alpha_t, \sigma_t^2/\alpha_t^2)$ is equivalent to $g \circ X^{(1:L)} \sim \bar{\mathcal{N}}(g \circ X_t/\alpha_t, \sigma_t^2/\alpha_t^2)$ and the target density $p_d(X)$ is G-invariant, we have

$$\begin{aligned} X^{*\prime} &= g \circ X^{(l_*)} \\ l_* &\sim \text{Categorical}\left(\frac{\tilde{p}_d(X^{(1)})}{\sum_{l=1}^{L} \tilde{p}_d(X^{(l)})}, \cdots, \frac{\tilde{p}_d(X^{(L)})}{\sum_{l=1}^{L} \tilde{p}_d(X^{(l)})} \right), \\ X^{(1:L)} &\sim \bar{\mathcal{N}}(X_t/\alpha_t, \sigma_t^2/\alpha_t^2). \end{aligned} \tag{D.74}$$

Comparing Equation (D.72) and Equation (D.74), we conclude that $X^{*\prime} = g \circ X^*$, and hence the sample obtained by SIR is G-equivariant to X_t. Additionally, it is easy to see that the score at the sample obtained by SIR is also G-equivariant to X_t.

Hamiltonian Monte Carlo

If we use Hamiltonian Monte Carlo (HMC) to draw samples from $p_d(X|X_t)$ to estimate Equation (D.49), we need to make the following two mild assumptions.

1. The initial state X of HMC should be zero-centred and G-equivariant with respect to X_t. This can be easily achieved, as the initial state can simply be a sample from the proposal $q(X|X_t) = \bar{\mathcal{N}}(X|X_t/\alpha_t, \sigma_t^2/\alpha_t^2)$ used in the IS estimator, or can be a sample from SIR.
2. The momentum variable V in HMC should follow a zero-centred and G-invariant Gaussian distribution, which automatically holds true for common choices such as standard Gaussian distribution and isotropic Gaussian distribution.

Under these assumptions, we can show that the samples generated by HMC are zero-centred and G-equivariant to X_t. Below, we show that this holds for the first HMC step. The remaining steps can be simply proved recursively by viewing the sample from the previous HMC step as the initial state for the current HMC step.

We first note that the score of the denoising posterior $p_d(X|X_t)$,

$$\nabla_X \log p_d(X|X_t) = \nabla_X \log p_d(X) + \nabla_X \log \bar{\mathcal{N}}(X_t|X), \tag{D.75}$$

is zero-centred and G-equivariant if both X and X_t are zero-centred. This is because $\nabla_X \log p_d(X)$ is zero-centred as shown in Proposition D.4.1 and G-equivariant, and

$$\nabla_X \log \bar{\mathcal{N}}(X_t|X) \propto X - X_t \tag{D.76}$$

is also zero-centred and G-equivariant. For one HMC step with the leapfrog algorithm, we have

$$\begin{aligned} V &\sim \bar{\mathcal{N}}(O, m), \\ V_{\eta/2} &= V + \frac{\eta}{2} \nabla_X \log p_d(X|X_t), \\ X' &= X + \frac{\eta}{m} V_{\eta/2}, \\ V' &= V_{\eta/2} + \frac{\eta}{2} \nabla_{X'} \log p_d(X'|X_t). \end{aligned} \tag{D.77}$$

The proposed sample X' will be accepted according to the Metropolis-Hastings algorithm:

$$\begin{aligned}\alpha_{\text{HMC}} &= \min\left\{1, \frac{\bar{\mathcal{N}}(V'|O,m)p_d(X'|X_t)}{\bar{\mathcal{N}}(V|O,m)p_d(X|X_t)}\right\} \\ &= \min\left\{1, \frac{\bar{\mathcal{N}}(V'|O,m)\tilde{p}_d(X')\bar{\mathcal{N}}(X_t|X')}{\bar{\mathcal{N}}(V|O,m)\tilde{p}_d(X)\bar{\mathcal{N}}(X_t|X)}\right\}.\end{aligned} \tag{D.78}$$

It is easy to see that all operations maintain the zero-centred property of V and X, and therefore V' and X' are zero-centred.

Also, it is easy to check that α_{HMC} is G-invariant. Hence, we will only focus on the leapfrog step:

$$X' = X + \frac{\eta}{m}V + \frac{\eta^2}{2m}\nabla_X \log p_d(X|X_t), \quad V \sim \bar{\mathcal{N}}(O,m). \tag{D.79}$$

Suppose that we apply $g \in G$ to X_t and therefore X (as they are G-equivariant by assumption). Now, the leapfrog step becomes:

$$X'' = g\circ X + \frac{\eta}{m}\left(g\circ V + \frac{\eta}{2m}\nabla_{g\circ X}\log p_d(g\circ X|g\circ X_t)\right), \quad g\circ V \sim \bar{\mathcal{N}}(O,m) \tag{D.80}$$

$$= g\circ\left(X + \frac{\eta}{m}V + \frac{\eta^2}{2m}\nabla_X \log p_d(X|X_t)\right), \quad V \sim \bar{\mathcal{N}}(O,m) \tag{D.81}$$

$$= g\circ X'. \tag{D.82}$$

This shows that an HMC leapfrog step X' is G-equivariant. It is easy to show that V' is also G-equivariant using the same arguments.

Langevin Dynamics

We can also use Langevin dynamics (e.g., ULA or MALA) to draw samples from $p_d(X|X_t)$ to estimate Equation (D.49). Note that MALA is just ULA with an additional Metropolis-Hastings correction step. Using the same argument as that for HMC, it is easy to check that the acceptance rate α_{MALA} is G-invariant. Therefore, it suffices to investigate a single step of ULA update. We will make two mild assumptions similar to those for HMC.

1. The initial state X of ULA should be zero-centred and G-equivariant with respect to X_t. This can be easily achieved in a similar way as discussed for HMC above.
2. The Brownian motion used in ULA should be zero-centred and G-equivariant with respect to X_t, which automatically holds true for standard Gaussian noise.

For a matrix of zero-centred standard Gaussian noise $\mathcal{E} \sim \bar{\mathcal{N}}(O, 1)$, it is easy to see that the ULA update X' is zero-centred from the Langevin equation:

$$X' = X + \eta \nabla_X \log p_d(X|X_t) + \sqrt{2\eta}\mathcal{E}, \tag{D.83}$$

since we have shown that $\nabla_X \log p_d(X|X_t)$ is zero-centred.

Suppose that we apply $g \in G$ to X_t and therefore X (as they are G-equivariant by assumption). Noticing that $\mathcal{E}$ is G-invariant and using the same arguments as those for HMC, we can show that the ULA update becomes

$$\begin{aligned} X'' &= g \circ X + \eta \nabla_{g\circ X} \log p_d(g \circ X|g \circ X_t) + \sqrt{2\eta}(g \circ \mathcal{E}), \quad g \circ \mathcal{E} \sim \bar{\mathcal{N}}(0, 1) && \text{(D.84)} \\ &= g \circ \left(X + \eta \nabla_X \log p_d(X|X_t) + \sqrt{2\eta}\mathcal{E}\right), \quad \mathcal{E} \sim \bar{\mathcal{N}}(0, 1) && \text{(D.85)} \\ &= g \circ X'. && \text{(D.86)} \end{aligned}$$

This shows that a ULA update X' is G-equivariant

Annealed Importance Sampling

Another choice for drawing samples from $p_d(X|X_t)$ to estimate Equation (D.49) is annealed importance sampling (AIS) or AIS followed by SIR. Recall that in IS, we directly draw samples from a proposal $q(X|X_t) = \bar{\mathcal{N}}(X|X_t/\alpha_t, \sigma_t^2/\alpha_t^2)$ to target at the denoising posterior distribution $p_d(X|X_t) \propto p_d(X)\bar{\mathcal{N}}(X_t|\alpha_t X, \sigma_t^2)$. AIS introduces a sequence of intermediate distributions $\{\pi_{(k)}(X)\}_{k=1}^K$ that interpolate between the proposal $q(X|X_t)$ and the denoising posterior $p_d(X|X_t)$ to facilitate a smoother transition:

$$\begin{aligned} \pi_{(k)}(X) &\propto q(X|X_t)^{1-\beta_k} p_d(X|X_t)^{\beta_k} && \text{(D.87)} \\ &\propto \left(\bar{\mathcal{N}}(X|X_t/\alpha_t, \sigma_t^2/\alpha_t^2)\right)^{1-\beta_k} \left(p_d(X)\bar{\mathcal{N}}(X_t|\alpha_t X, \sigma_t^2)\right)^{\beta_k} && \text{(D.88)} \\ &\propto \tilde{p}_d(X)^{\beta_k} \bar{\mathcal{N}}(X_t|\alpha_t X, \sigma_t^2), && \text{(D.89)} \end{aligned}$$

where $0 = \beta_0 < \beta_1 < \cdots < \beta_{K-1} < \beta_K = 1$, $\pi_{(0)}(X) \coloneqq q(X|X_t)$ is the proposal distribution, and $\pi_{(K)}(X) \coloneqq p_d(X|X_t)$ is the denoising posterior distribution.

The AIS algorithm runs iteratively as follows:

- Draw $X_{(0)} \sim \pi_{(0)} \coloneqq q$ from the proposal distribution.
- For $k = 1, 2, \ldots, K-1$, draw $X_{(k)} \sim \pi_{(k)}$ by running an MCMC sampler (e.g., HMC) whose initial state is the previous sample $X_{(k-1)}$ and which leaves the current intermediate distribution $\pi_{(k)}$ invariant.

In the end, AIS produces a sample $X \coloneqq X_{(k-1)}$ which is close to $\pi_{(K)}(X) = p_d(X|X_t)$. We can then calculate the IS weight in the joint space over $X_{(0:K-1)}$, which is the AIS weight:

$$\tilde{w}_{\text{AIS}}(X_{(0:K-1)}) = \frac{\pi_{(1)}(X_{(0)})}{\pi_{(0)}(X_{(0)})} \frac{\pi_{(2)}(X_{(1)})}{\pi_{(1)}(X_{(1)})} \cdots \frac{\pi_{(K)}(X_{(K-1)})}{\pi_{(K-1)}(X_{(K-1)})} \tag{D.90}$$

$$\propto \prod_{k=1}^{K} \tilde{p}_d(X_{(k-1)})^{\beta_k - \beta_{k-1}}. \tag{D.91}$$

Formally, the (self-normalised) AIS estimator works as follows:

$$\begin{aligned} &\int (\alpha_t(X + \nabla \log p_d(X)) - X_t) p_d(X|X_t) dx \\ &\qquad \approx \alpha_t \frac{\sum_{l=1}^{L} (X^{(l)} + \nabla \log p_d(X^{(l)})) \tilde{w}_{\text{AIS}}(X^{(l)}_{(0:K-1)})}{\sum_{l'=1}^{L} \tilde{w}_{\text{AIS}}(X^{(l')}_{(0:K-1)})} - X_t, \end{aligned} \tag{D.92}$$

where $X^{(1:L)}_{(0:K-1)} \sim$ AIS.

Assume that we use HMC, MALA or ULA as the transition kernel in AIS with their respective assumptions. Then, the sequence of AIS samples $X^{(1:L)}_{(0:K-1)}$ will be G-equivariant to X_t. Additionally, we note that $\tilde{w}_{\text{AIS}}$ is G-invariant to X, since $p_d(X)$ and $(\tilde{p}_d(X))$ is G-invariant to X. Also, the score $\nabla \log p_d(X)$ is G-equivariant to X. We can now conclude that the AIS estimator for MSI is G-equivariant to X_t following the same arguments as those for IS.

Finally, if we perform AIS followed by SIR, the resulting estimator will also be G-equivariant to X_t following the same arguments as those for SIR with IS.

D.5 Experimental Setup

D.5.1 Mixture-of-Gaussians

We train all methods for 2.5 hours, which allows all of them to converge.

For our approach, we choose the total number of diffusion steps to be $T = 30$. We use a variance-preserving (VP) scheme with $\alpha_t = \prod_{s=1}^{t}(1 - \beta_s)$ (Ho et al., 2020) and a linear schedule with β_t ranging from 10^{-4} to 0.7. We choose the weighting function to be $w(t) = 1/\alpha_t$. For the score network $s_\phi(x_t)$, we use a 5-layer MLP with a hidden dimension of 400 and SiLU activation. In each inner loop, we use Adam to train the score network $s_\phi(x_t)$ for 50 iterations using DSM with a learning rate of 10^{-4} and a batch size of $1,024$. For the neural sampler $f_\theta(z)$, we use a 5-layer MLP with a latent dimension of 2, a hidden dimension of 400 and SiLU activation. We use Adam to train the neural sampler $f_\theta(z)$ using MSI with a

learning rate of 10^{-3}, a batch size of $1,024$, and gradient norm clip 10.0. Regarding posterior sampling, we use AIS with 10 samples and 15 intermediate steps. For each AIS intermediate step, we use the HMC transition kernel with 1 leapfrog step and step size 1.0. We resample one of those 10 AIS samples according to the AIS weights and use that sample as the initial state for 5 steps of MALA with step size 10^{-2}. In the end, we estimate the MSI using this single sample (i.e., we perform Monte Carlo estimation with one sample).

For R-KL-based approaches, we align their experiment setups with that for our approach for a fair comparison. Specifically, we use a 5-layer MLP with a hidden dimension of 400 and SiLU activation for the score network $s_\phi(x_t)$. In each inner loop, we use Adam to train the score network $s_\phi(x_t)$ for 50 iterations using DSM with a learning rate of 10^{-4} and a batch size of $1,024$. For the neural sampler $f_\theta(z)$, we use a 5-layer MLP with a latent dimension of 2, a hidden dimension of 400 and SiLU activation. We use Adam to train the neural sampler $f_\theta(z)$ with a learning rate of 10^{-3}, a batch size of $1,024$, and gradient norm clip 10.0.

For FAB (Midgley et al., 2023) and iDEM (Akhound-Sadegh et al., 2024), we use exactly the same setups as described in the respective papers. Note that both of them use replay buffers to balance exploration and exploitation. In addition, all methods except iDEM work under the original scale $[-50, 50]$ of the target distribution. In contrast, iDEM normalises the target to the range $[-1, 1]$, which may simplify the task.

D.5.2 Many-Well-32

We train all compared models until convergence. Training and sampling time for each model can be found in Table 5.3.

For our approach, we choose the total number of diffusion steps to be $T = 30$. We use a variance-preserving (VP) scheme with $\alpha_t = \prod_{s=1}^{t}(1 - \beta_s)$ (Ho et al., 2020) and a linear schedule with β_t ranging from 10^{-4} to 0.15. We choose the weighting function to be $w(t) = 1/\alpha_t$. For the score network $s_\phi(x_t)$, we use a 5-layer MLP with a hidden dimension of 400 and SiLU activation. In each inner loop, we use Adam to train the score network $s_\phi(x_t)$ for 50 iterations using DSM with a learning rate of 10^{-4} and a batch size of $1,024$. For the neural sampler $f_\theta(z)$, we use a 5-layer MLP with a latent dimension of 32, a hidden dimension of 400 and SiLU activation. We use Adam to train the neural sampler $f_\theta(z)$ using MSI with a learning rate of 10^{-3}, a batch size of $1,024$, and gradient norm clip 10.0. Regarding posterior sampling, we use AIS with 10 importance samples and 15 intermediate steps. For each AIS intermediate step, we use the HMC transition kernel with 1 leapfrog step and step size 0.3. We resample one of those 10 AIS samples according to the AIS weights and use that sample as the initial state for 5 steps of MALA with step size 5×10^{-2}. In the end, we estimate the MSI using this single sample.

For R-KL-based approaches, we align their experiment setups with that for our approach for a fair comparison. Specifically, we use a 5-layer MLP with a hidden dimension of 400 and SiLU activation for the score network $s_\phi(x_t)$. In each inner loop, we use Adam to train the score network $s_\phi(x_t)$ for 50 iterations using DSM with a learning rate of 10^{-4} and a batch size of $1,024$. For the neural sampler $f_\theta(z)$, we use a 5-layer MLP with a latent dimension of 32, a hidden dimension of 400 and SiLU activation. We use Adam to train the neural sampler $f_\theta(z)$ with a learning rate of 10^{-3}, a batch size of $1,024$, and gradient norm clip 10.0.

For FAB, we use exactly the same setup as described in Midgley et al. (2023). For iDEM, we use the same setup as that for experiments in internal coordinates as described in (Akhound-Sadegh et al., 2024) but change the maximum score norm clip threshold to $1,000$, increase the number of MC samples to $1,000$, and reduce σ_{max} in the noise schedule to 1.0, since the default setting does not work on this target distribution at all. Note that both of FAB and iDEM use replay buffers to balance exploration and exploitation.

D.5.3 Double-Well-4

We train all compared models until convergence. Training and sampling time for each model can be found in Table 5.3.

For our approach, we choose the total number of diffusion steps $T = 30$. We use a variance-preserving (VP) scheme with $\alpha_t = \prod_{s=1}^{t}(1 - \beta_s)$ (Ho et al., 2020) and a linear schedule with β_t ranging from 10^{-6} to 0.05. We found that using a constant weighting function, $w(t) = 1$, is beneficial for handling these complex targets. We use EGNN following Hoogeboom et al. (2022) for both the score network s_ϕ and the neural sampler f_θ. The neural sampler has 8 layers with a hidden dimension of 144 and ReLU activation. The score network has 4 layers with the same width, and it is additionally conditioned on t. We use Adam to train the score network s_ϕ for 100 iterations using DSM with a learning rate of 10^{-4} and a batch size of 1024. We use Adam to train the neural sampler $f_\theta(z)$ using MSI with a learning rate of 5×10^{-4}, a batch size of 1024, and gradient norm clip 10.0. Regarding posterior sampling, we use AIS with 20 importance samples and 10 intermediate steps. For each AIS intermediate step, we use 1-step MALA transition kernel with step size 0.01. We resample one of those 20 AIS samples according to the AIS weights and use that sample as the initial state for 50 steps of MALA. We dynamically adjust the MALA step size to maintain an acceptance rate between 0.5 and 0.6. Specifically, we increase the step size by a factor of 1.5 when the acceptance rate exceeds 0.6 and decrease it by a factor of 1.5 when the acceptance rate drops below 0.5. In the end, we estimate the MSI using this single sample.

Additionally, we employ early stopping during training. Specifically, we generate 2,000 samples using the neural sampler, which serve as the *predictions*. These samples are then

used as the initial states for 50 MALA steps, targeting the target energy. The 2,000 samples obtained after MALA are treated as the *validation* set. We evaluate the energy of both the predictions and the validation set, then calculate the total variation distances (TVDs) between their energy histograms. We save the model with the lowest such TVD. This criterion can be interpreted as asking the following question: how much improvement can be achieved by running a small number of Langevin steps to the model samples? The less improvement we can achieve, the better the model samples are.

For FAB (Midgley et al., 2023) and iDEM (Akhound-Sadegh et al., 2024), we use exactly the same setups as described in the respective papers. Note that both of them use replay buffers to balance exploration and exploitation.

D.5.4 Lennard-Johns-13

We train all compared models until convergence. Training and sampling time for each model can be found in Table 5.3.

For our approach, we choose the total number of diffusion steps $T = 30$. We use a variance-preserving (VP) scheme with $\alpha_t = \prod_{s=1}^{t}(1 - \beta_s)$ (Ho et al., 2020) and a linear schedule with β_t ranging from 10^{-6} to 0.05. We also use a constant weighting function, $w(t) = 1$. Both the score network s_ϕ and the neural sampler network f_θ share the same 8-layer architecture with a hidden dimension of 192 and ReLU activation. We use Adam to train the score network s_ϕ for 100 iterations using DSM with a learning rate of 10^{-4} and a batch size of 256. We use Adam to train the neural sampler f_θ using MSI with a learning rate of 5×10^{-4}, a batch size of 256, and gradient norm clip 10.0. Regarding posterior sampling, we use IS with 500 importance samples. We then resample one of those IS samples according to the IS weights and use that sample as the initial state for 1,000 steps of MALA. We also dynamically adjust the MALA step size to maintain an acceptance rate between 0.5 and 0.6. Specifically, we increase the step size by a factor of 1.5 when the acceptance rate exceeds 0.6 and decrease it by a factor of 1.5 when the acceptance rate drops below 0.5. Unlike previous tasks, we found that using only the last sample from MALA sometimes leads to suboptimal performance. To improve stability, we track the samples and their gradients from the last 500 steps of MALA, and estimate the MSI using those 500 samples. It is important to note that since the samples and their scores are already computed during MALA, using more samples in the Monte Carlo estimator does not incur any additional computational costs. We note that smoothing the LJ target following Moore et al. (2024) can help to stabilise the training. However, our approach works well even without this smoothing. Additionally, we employ early stopping in the same way as in DW-4.

For FAB (Midgley et al., 2023) and iDEM (Akhound-Sadegh et al., 2024), we use exactly the same setups as described in the respective papers. Note that both of them use replay buffers to balance exploration and exploitation.

D.6 Guidance for Hyperparameter Tuning

Below, we provide guidance for tuning the key hyperparameters in our method.

- β_T in the VP noise schedule should not be too large, as this can lead to inaccurate posterior sampling and worse performance. This is because the neural sampling will tend to favour the mean of the global mass for large β, which can be seen in Figure 5.3 in the main text: the model is biased towards the mean as the noise level increases.
- For DW-4 and L-13, it is important to use a large EGNN for the neural sampler. Unlike diffusion models which generates each sample with a large number of sampling steps (Akhound-Sadegh et al., 2024; Hoogeboom et al., 2022), our approach involves learning a one-step sampling generator and thus requires greater model capacity. We tested smaller networks, such as an EGNN with 6 layers and a hidden dimension of 128, which resulted in worse performance compared to the larger architecture used in our experiments. On the other hand, the score network does not need to have the same capacity. For example, we found that using a shallower score network is sufficient to achieve optimal performance on DW-4. Additionally, we found using ReLU in EGNNs resulted in better performance than SiLU, possibly because ReLU allows our one-step neural sampler to model complex distributions with large Lipschitz constants.
- The weight function $w(t)$ also requires careful tuning. While other weighting functions commonly used in diffusion models include σ_t^2/α_t or σ_t^2/α_t^2, we found using $1/\alpha_t$ or even uniform weighting is more stable in our approach. An empirical guideline for choosing between these is as follows. For more complex targets like DW-4 and LJ-13, a uniform weighting function encourages exploitation. On the other hand, for highly multi-modal target distributions like MW-32, using the $1/\alpha_t$ weighting function can accelerate exploration.
- The batch size cannot be too small. Empirically, we found that training could be unstable if a small batch size is used. In general, we recommend using a large batch size like 1,024 if it fits into the GPU memory.

D.7 Reproducibility Statement

Our open-source implementation of R-DiKL for reproducing the experimental results in Chapter 5 can be found at https://github.com/jiajunhe98/DiKL. In our experiments, we also used the following codebases for the benchmarks and baselines.

- The implementation of the DW-4 and LJ-13 energy functions was taken from the bgflow repository (https://github.com/noegroup/bgflow) (Köhler et al., 2020).
- The PyTorch implementation of FAB (https://github.com/lollcat/fab-torch) (Midgley et al., 2023) was used for its experiments on MoG-40 and MW-32.
- The JAX implementation of FAB (https://github.com/lollcat/se3-augmented-coupling-flows) (Midgley et al., 2024) was used for its experiments on DW-4 and LJ-13.
- The PyTorch implementation of iDEM (https://github.com/jarridrb/DEM) (Akhound-Sadegh et al., 2024) was used for all its experiments.