THE UNIVERSITY
of EDINBURGH

Sampling Statistical Distributions in Physics: A Machine Learning Approach

Michael Wilson

Doctor of Philosophy
The University of Edinburgh
October 2021

Abstract

This thesis presents work which uses Machine Learning techniques in a variety of sampling situations which appear in physics. In the first Chapter some background on Machine Learning will be presented which will lay the foundations required for the later Chapters.

Next we will look at how a specific Machine Learning model, the Restricted Boltzmann Machine, can be trained to approximate a target distribution from data which has already been sampled from the target distribution. We estimate observables on states sampled from trained models and compare them to observables estimated directly from the training data. We present a technique for estimating the likelihood function of the model using annealed importance sampling. Finally we present a closed form expression for extracting the N-point interactions which the model learns from the data directly from the parameters of the model, a result which is useful for a range of fields which study binary data.

In the next Chapter we investigate a different generative model, the normalizing flow, and investigate its efficacy of generating configurations for a lattice scalar field theory. An initial study which quantifies how the cost of training this model scales with the system size is performed. Whilst the cost of training our models is significantly less than those reported in the proof of principle study which first presented using these models for this purpose [1], we discuss how there is still an exponential scaling of the training cost with the system size which must be overcome in order for these models to be practically useful.

Finally we investigate inverse problems from a Bayesian perspective. With this framework, we are faced with the task of sampling from the posterior distributions in model space given the data. An approach for sampling in model space presented by the NNPDF collaboration is examined within this formal framework. We present some statistical estimators which can be used to validate a methodology which

produces a sample of models. These estimators can be implemented in a closure test [2], where the data is artificially generated from a pre-existing underlying law. We show how these estimators can be used to check that the model distribution is self consistent when the data is fluctuated according to its prior. A proof of principle example is presented by performing a closure test using the latest NNPDF methodology and we show that the NNPDF MC approach is successful at producing a sample of model replicas which have faithful uncertainties. Whilst these estimators are practical to implement and are shown to be useful in a non-trivial setting, we discuss the possibility of defining some estimators directly in model space which could give more general information on model uncertainties for a wide range of inverse problems.

Declaration

I declare that this thesis was composed by myself, and is a record of work performed as part of the NNPDF collaboration or separate collaborations with Guido Cossu, Tommaso Giani, Ava Khamseh, Joe Marsh Rossney and my supervisor Luigi Del Debbio. Unless explicitly stated, the results presented in this thesis are either mine, or the product of collaboration in which I have made a significant contribution.

The work in Chapter 2 is based on work published in [3]

The work in Chapter 3 is based on work published in [4]

The work in Chapter 4 is based on work published in [5], with some results published in [6]. The analysis tools, used to produce the results, are published in [7].

(*Michael Wilson, October 2021*)

Acknowledgements

During my time in Edinburgh I've interacted with many brilliant people. Firstly, I would like to thank my supervisor Luigi Del Debbio for the support, stimulating discussions and providing opportunities to do interesting and exciting research.

I'd also like the thank all members of the NNPDF collaboration for creating a brilliant and supportive environment within which I was able to develop myself personally, academically and professionally and contribute to some amazing work with some exceptional researchers. For their consistently useful advice, fantastic work ethic and thought provoking discussions I am greatly appreciative. I am extremely grateful for the guidance of some of the more senior members of the collaboration, in particular Richard Ball, Stefano Carazza, Stefano Forte, Juan Rojo and Maria Ubiali. I'd also like to specially mention the following individuals who I worked with on a day to day basis, and were able to teach me technical skills, provide emotional support and unblock me when I was stuck: Juan Cruz-Martinez, Shayan Iranipour, Zahari Kassabov, Emanuele Nocera and Cameron Voisey. Furthermore, I'd like to thank Tommaso Giani and Rosalyn Pearson who not only were fantastic colleagues in NNPDF, but also shared the experience of being PhD students in Edinburgh. Whilst the events of the last couple of years prevented us from meeting up in person, the camaraderie which formed between us was invaluable, and I will always remember the fun we had, especially when we shared an office in our third year.

Outside of NNPDF, I'm also grateful to other members of the particle theory group at Edinburgh. Most notably were my collaborators Guido Cossu, Ava Khamseh and Joe Marsh Rossney. Along the course of my PhD I've also had the pleasure of befriending other students outside of particle theory: James Aston, Lewis Conway, Pete Cooke, Oscar Hall, Conor Hamill, and Ashley Tattersall. I'd like to thank them for helping create many fun memories during my time here in Edinburgh.

At the end of my third year, I was lucky enough to get the opportunity to undertake an internship at FreeAgent. I'd like to thank everyone there for making the experience extremely useful and in particular David Edwards, Dave Evans and Owen Turner for the huge amount of support and advice during my time there.

The support of my family can't be understated, and I'd particularly like to thank

my parents for providing support through all my endeavours. Finally, I'd like to thank my partner Rachel whose support is immeasurable, this thesis is dedicated to you.

Contents

List of Figures

List of Tables

Introduction

Machine learning (ML) has become a ubiquitous term over the last decade in both industry and science. The typical machine learning task involves making some kind of statistical inference of some model parameters from observed data. In many areas of physics this kind of approach easily out-dates the zeitgeist of ML or artificial intelligence (AI), but there are still physics communities which are, rightfully, sceptical of this current trend of integrating machine learning techniques into well established tool-chains.

The field of ML has emerged from the field of AI. For most people the term AI probably is associated with ideas and themes which come from science fiction - sentient beings which share many traits with humans. In fact, the conception of certain tools which are used today in AI/ML, such as the artificial neural networks, were motivated by nature [12], however many applications of AI which we interact with on a day to day basis are far less ambitious in scope than creating a new form of life. The distinction between AI and ML is that whilst AI describes the concept of designing systems which can make choices like humans, ML is a subset of techniques which allow parts of those systems to learn from data without being explicitly programmed with a set of rules.

Defined as such, the distinction between fitting ML model parameters from data and physicists fitting physical constants from experimental observations is not entirely clear. A broader definition of ML is probably better understood by outlining the major paradigms: supervised learning, unsupervised learning and reinforced learning. The next chapter will focus on defining these and any other concepts which will be used later on - with more emphasis on supervised and unsupervised learning which are the techniques used in the work this thesis is based upon.

ML techniques have become more accessible than ever before, especially with

the availability of many `python` libraries which combine the ease of high level programming with back-ends which can leverage the latest state-of-the-art high performance computing hardware such as Graphical Processing Units (GPUs) and more recently Tensor Processing Units (TPUs). The allure of being able to rapidly prototype a model which can be trained quickly with hardware acceleration and contains many parameters so can potentially model very complicated relationships between input features and output responses is clearly a feature of ML which has contributed to its popularity in industrial settings over the last few years, but it's clear in physics we must take a bit more care.

In theoretical physics in particular, it's perhaps not as clear how ML can be used. ML tools are often used as black boxes which, once the parameters of the model have been optimised using the available data, provide no insight into the actual problem. In theoretical physics, we are often interested in understanding *how* to solve the problem and we are less inspired by a brute force method which appears to give a reasonable answer.

In Chapter One of this thesis we shall provide some fundamental terms and concepts used in ML which will be necessary in order to progress to the work presented in subsequent chapters. Chapter Two is concerned with Restricted Boltzmann machines (RBMs), a class of generative models used in unsupervised learning, and how we can use our pre-existing knowledge of the Ising model to explore how RBMs work as well as deriving an expression for determining physical couplings of the modelled system directly from the model parameters. Chapter Three will focus on another class of generative models known as normalizing flows. This chapter will instead focus on whether these models can be used as part of a MC tool-chain to generate lattice field theory configurations, and in particular whether using these models is feasible and efficient enough to justify using a ML model over a more traditional sampling technique. Chapter Four will then shift focus over to the paradigm of supervised learning, where we will set out a Bayesian framework for solving inverse problems and derive some statistical estimators which investigate the faithfulness of uncertainties. The relationship between the posterior distribution in the space of models, from a Bayesian perspective, and the output of performing a typical fit to data using machine learning tools will be discussed. The combination of this formal framework and estimators are used to understand and validate the methodology which will be used to produce the NNPDF4.0 parton distribution functions (PDFs) and the result of evaluating the derived estimators in this context shall be investigated. Finally will we discuss

briefly how the generation of lattice configurations and the Bayesian inverse problems are conceptually similar problems to solve, despite being classified as unsupervised and supervised learning respectively.

Chapter 1

Fundamentals of Machine Learning

In this chapter we will review some fundamentals of machine learning which will lay the foundations for future Chapters. There are several reviews of the field of ML which are specifically aimed at physicists [13, 14]. This chapter will serve as a general overview, introducing terms which are common to ML literature and used in subsequent chapters.

Machine learning is usually split into three major paradigms: supervised learning, unsupervised learning and reinforcement learning. We will restrict ourselves to the former two paradigms, which the work presented in the later chapters fall under.

1.1 Supervised learning

In supervised learning we have N_{data} input data, $X_\mu \in \mathcal{R}^p$, where $\mu = 1, \ldots, N_{\text{data}}$ and a set of corresponding measured responses $y_\mu \in \mathcal{R}^d$. The data $\mathcal{D}$ is then given by the full set of input-output pairs

$$\mathcal{D} = \{(X_1, y_1), (X_2, y_2), \ldots, (X_{N_{\text{data}}}, y_{N_{\text{data}}})\} \ . \tag{1.1}$$

The next ingredient is a model, which is a function which acts upon the input data and has some parameters θ, which we denote as $\mathcal{G}(X_\mu; \theta)$. The *training* or *learning* task is then to find the set of parameters such that the model output is a good approximation of the output data. In practical terms this involves defining

a *loss* or *cost* function, $\mathcal{L}$, and then solving

$$\theta_* = \arg\min_{\theta} \sum_{\mu}^{N_{\text{data}}} \mathcal{L}\left(\mathcal{G}(X_\mu;\theta), y_\mu\right) \ , \tag{1.2}$$

typically using a numerical routine. The form of the loss function depends on the specific task, and in the later chapters the loss which is minimised will be discussed in detail. For now consider a concrete example of a loss function, the quadratic loss which appears in least squares methods

$$\mathcal{L}\left(\mathcal{G}(X_\mu;\theta), y_\mu\right) = \left(\mathcal{G}(X_\mu;\theta) - y_\mu\right)^2 \tag{1.3}$$

which clearly measures the euclidean distance between the model prediction and the measured value. The data which is used during the training is called the *training data*, it's very common in ML literature to split the available data into training and *test* data. The sum which appears in Eq. 1.2 then just runs over the training data. The test data is often reserved for assessing the model performance, specifically on data which was not included in the fit. The motivation behind this is to test the ability of the trained model to *generalise* to new data, since typical usage of models trained in this way is to make predictions about data for which the output data, y is unknown.

For complicated models, finding the optimal parameters can be highly non-trivial, because the loss function might be non-convex in the model parameters. Recently, the most popular approaches often involve choosing a model and loss function such that the loss is differentiable with respect to the model parameters. The model parameters can then be updated using a gradient-based algorithm. Consider the simplest gradient descent (GD) algorithm

$$\begin{aligned} v^t &= \nabla_\theta \sum_{\mu}^{N_{\text{data}}} \mathcal{L}\left(\mathcal{G}(X_\mu;\theta), y_\mu\right)\Big|_{\theta=\theta^t} \\ \theta^{t+1} &= \theta^t - \alpha v^t \ , \end{aligned} \tag{1.4}$$

where v^t is the gradient of the sum of the loss evaluated on the training data, with parameters at a timestep or *epoch* t and α is the learning rate. For a small enough learning rate we can guarantee that GD will converge to a local minimum of the loss function. As mentioned, the loss function might be non-convex however and so the parameters obtained from GD will depend on the initialisation of the parameters. Furthermore, if the learning rate is too low then GD will take a long

time to converge, but if it is too high then the algorithm may never converge, and instead keep "missing" the local minimum of the loss function. The latter point is an example of a *hyperparameter* choice, a parameter which must be chosen in advance of the training and dictates the procedure by which the model parameters are found. The former is a bigger issue, and is one of the reasons the simple GD algorithm is not used. Other issues include the fact that gradients are expensive to compute for large datasets and that GD can get stuck on saddle points of the loss function. Addressing these issues leads to the algorithm which is the basis for many popular minimisation algorithms: Stochastic Gradient Descent (SGD).

SGD overcomes some of the shortcomings of GD by splitting the total training dataset into *batches* or *minibatches*, small subsets which can be much more manageable from a computational perspective. In some ML literature, batch might refer to the full dataset and minibatch may refer to one of these subsets, however throughout this thesis the terms will be used interchangeably unless stated otherwise. The stochasticity of SGD is introduced by randomly splitting the dataset into minibatches, it's also common to randomly shuffle the way the minibatches are split each epoch, which now refers to the step within which all training data is presented to the model. The parameter update is otherwise identical to GD, except now the data is split into N_{batch} batches B_k and the gradient of the loss is given by

$$v^t = \nabla_\theta \sum_{\mu \in B_k} \mathcal{L}\left(\mathcal{G}(X_\mu;\theta), y_\mu\right)\Big|_{\theta=\theta^t} , \tag{1.5}$$

the only difference between Eq. 1.5 and the gradient defined in Eq. 1.2 is the set of data over which the gradient is evaluated. In this scheme, the parameters get updated N_{batch} times every epoch. Whilst SGD already offers some improvements the simple GD algorithm, there are further variants of this algorithm which aim to improve the efficiency. One such variation involves adding in a momentum term [15], the gradient term is replaced by

$$v^t = \gamma v^{t-1} + \nabla_\theta E(\theta^t) , \tag{1.6}$$

where we introduced the momentum term $0 \leq \gamma \leq 1$ and the shorthand notation of $\nabla_\theta E(\theta^t) = \nabla_\theta \sum_{\mu \in B_k} \mathcal{L}\left(\mathcal{G}(X_\mu;\theta), y_\mu\right)\Big|_{\theta=\theta^t}$ where the gradient of the loss is implicitly evaluated over a minibatch. The role of the momentum term is to introduce a running average of the most recent previous gradients, with more historic gradients gaining a diminishing weight. The idea of momentum is to

increase the rate of change of parameters in directions with small but persistent gradients and to suppress oscillations in directions where the gradients are constantly changing. Empirical studies have shown the importance of a carefully chosen momentum term [16].

Further variants of SGD exist which also take into account the second moment of the gradient such as Adadelta, Adam and RMSprop. For example, the update rule of RMSprop is given by

$$\begin{aligned} v^t &= \nabla_\theta E(\theta^t) \\ s^t &= \beta s^{t-1} + (1-\beta) {v^t}^2 \\ \theta^{t+1} &= \theta^t - \alpha \frac{v^t}{\sqrt{s^t - \epsilon}} \ , \end{aligned} \tag{1.7}$$

where β is yet another hyperparameter, which controls the characteristic time of the second moment in a similar manner to the momentum γ, and ϵ is simply a regularisation term to avoid dividing by zero. The idea of this algorithm is to allow for larger learning rates in flat directions, whilst keeping the updates under control for parameters whose updates are consistently large.

It's not always clear which algorithm will produce the best results and the choice of minimisation algorithm is clearly linked to the specific landscape of the loss function for the data the model is being trained on, as well as the choices of other hyperparameters. An unsatisfactory feature of ML literature is that it can be difficult in differentiating between the effects of implementation details versus larger methodological changes when looking at the differences in results. Use of gradient based algorithms has been enabled by popular machine learning libraries such TensorFlow [17] and Pytorch [18] which automatically track operations performed between the input data and the loss and accumulate gradients along the way.

Note that gradient based algorithms are not the only available option, for example one can used genetic algorithms which are inspired by natural selection. These algorithms generate mutations of the current parameters and then choose the next parameters by picking the mutation which minimises the loss. These algorithms typically still have a choice of several hyperparameters such as learning rate, number of mutations and number of mutants. Whilst genetic algorithms forego the expensive gradient estimation, they require evaluating the loss for each mutant set of parameters which can still be rather expensive. As with all choices related

to hyperparameters, whether or not to use a gradient-based algorithm or a genetic algorithm is completely problem specific. From a practical standpoint there appears to be less attention on genetic algorithms, which means there is a smaller number of well maintained open source projects which can be incorporated into your own work.

1.2 Neural networks

In many ML applications, the model will involve some kind of neural network. Fig. 1.1 is an example of a small fully connected network, with a single hidden layer.

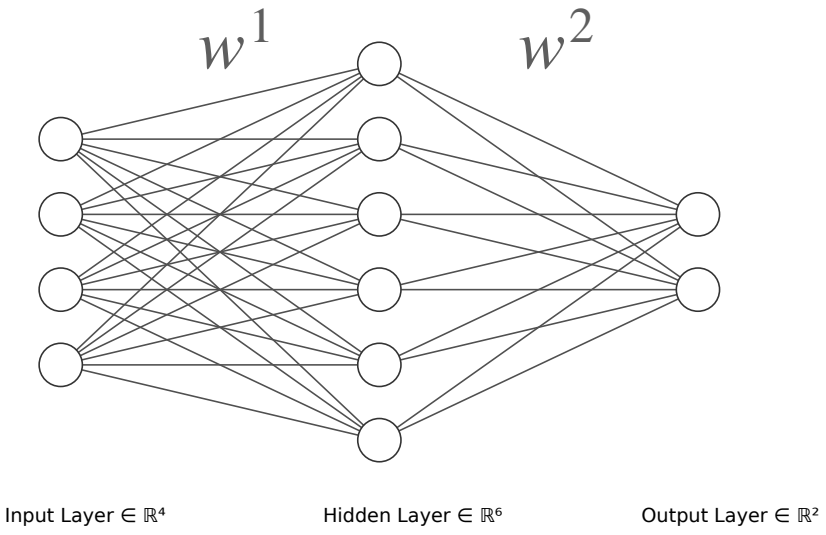

Figure 1.1 *Representing a fully connected neural network as a graph. The nodes represent inputs and outputs of each layer and are represented by vectors each element corresponding to the value of a node, x. The edges represent the weights of the network. Typically the weights of each layer are represented as an $N_{\text{in}} \times N_{\text{out}}$ matrix, w_l, where l is the layer index. The biases for each node are not explicitly shown in the figure and can be represented by a vector, b_l. Alternatively the biases can included by adding an additional node to the previous layer, whose value is always 1, then the biases are implicitly part of the weight matrix. With the former notation, the output of a layer is given by $g_l(w_l^T x + b_l)$, where g is the activation function which is often non-linear.*

The fully connected neural network, like that shown in Fig. 1.1, involves successively

applying a linear transformation followed by an activation function on the input. A single linear transformation followed the non-linear activation function associated with layer l is given by

$$x_l = g_l(b_l + w_l^T x_{l-1}) \tag{1.8}$$

where x_{l-1} is the input into the l^{th} layer, w_l and b_l are the corresponding weight matrix and bias vector respectively and g_l is the activation function. The full transformation of the fully connected neural network is then given by

$$g_L(b_l + w_L^T g_{L-1}(\dots g_2(b_2 + w_2^T g_1(b_1 + w_1^T X)))) \ . \tag{1.9}$$

The inner layers of the neural network which are not directly interacted with are referred to as the hidden layers, and the input and output layers may be referred to as visible layers, this language is used heavily in Chapter Two when we look at the Restricted Boltzmann Machine. The architecture of the neural network which includes: the depth of the network; the activation function for each layer; and the shape of each hidden layer are all hyperparameters. In many applications the activation function of the final layer is restricted, for example if the output of the network is supposed to represent a probability in binary classification task then the final activation function will likely be the sigmoid function

$$g^L = \frac{1}{1 + e^{-x}} \ , \tag{1.10}$$

which guarantees that the final output is properly bounded, $g^L(x) \in [0, 1]$. There are many other choices for activation functions but typically they are characterised by the first derivative being defined for all possible values of the input, so that they can be used alongside gradient based algorithms.

There are other kinds of neural networks, for example convolutional neural networks which, as the name suggests, take the convolution of the input layer with filters rather than each layer being fully connected by weights to the next. Convolutional neural networks enforce a translational invariance in the transformation applied to the input and require fewer parameters than fully connected networks. It is clear that models with fewer parameters require less compute time to be trained for the same number of epochs using the same number of training examples, and in problems such as image processing or pattern recognition there are arguments for imposing translational invariance. The shared weight architecture which convolutional neural networks are based on was first proposed for pattern recognition in images [19].

1.3 Hyperparameter tuning

In many of the previous sections, the term hyperparameter has appeared many times. Choosing hyperparameters must be done in advance of training a model, a hyperparameter is any parameter upon which the training process depends and is not restricted to the optimisation algorithm but also includes many aspects of the model architecture as well. There are no general rules for choosing the correct hyperparameters. In the work presented in later chapters, the process of choosing hyperparameters has often fallen to trial and error. Each time a model is trained, we measure some statistical estimators which we use to try and diagnose why the model training has plateaued. There are more sophisticated processes for tuning the hyperparameters, aptly called hyperparameter scans the basic principle is to choose a metric in advance and then algorithmically train models with different hyperparameters in an outer loop. The hyperparameters of the model which performs best on the chosen metric are then used. Brute force methods such as grid searches or random searches can be used, here the cost of an expert who tunes the parameters using some domain knowledge is traded for the computational cost of training many models under the assumption that the computational cost is cheaper and possibly faster. There are more sophisticated algorithms which attempt to perform a bayesian optimisation of the hyperparameters, such as Tree of Parzen Estimators (TPE) [20]. The basic idea is to start with some prior distributions for each hyperparameter and then as models are trained, the priors are updated with density being added to regions of hyperparameter space whose trained models performed well. These algorithms are described by the authors as bayesian, since they update the prior with information from previous trials, but it's worth pointing out that the method of updating the prior, at least in TPE, is just a prescription. Having said that, in the empirical results presented in [20], the algorithm does appear to outperform random search and either outperform or match human performance.

Regardless of whether the hyperparameter scan is performed manually or algorithmically, an important choice is the metric which is used to discriminate between models. It might be tempting to use the training loss as the metric for tuning the hyperparameters, but in many cases this can lead to a set of hyperparameters which *over-train*, a term which generally descibes when a statistical model fits statistical fluctuations in the training set rather than the underlying mapping from input to output. Over-training is typically characterised

by poor generalisation to unseen data. One method to circumvent this issue is to reserve a subset of the data during the hyperparameter scan, and evaluate the performance metric on this unseen data. In this way models are preferred which generalise best to data not included in the training. Note that this still doesn't help if there is an unknown systematic error present in the entire set of training data.

Once the model and various hyperparameters have been chosen and the parameters have been optimised using the available training data, the performance of the model is often judged using some kind of error function. Similarly to the loss, the error function heavily depends on the particular problem. In Chapter Four, the concept of the error function will be revisited in the context of inverse problems. For now one can consider reserving some of the available data as a test set and the error function could be the loss function evaluated on the test data set. This would be an example of some kind of generalisation error.

1.4 Unsupervised learning

In contrast to supervised learning, in unsupervised learning the data is just available as a set of input features X, there are no predefined labels or output responses and the task is to try and learn some kind of pattern directly from the input data. A popular example of unsupervised learning is clustering [21–23], which is similar to classification except the data has no predefined label and instead the data is grouped according to some kind of similarity metric. Clustering has been applied extensively in data mining [24], but also in fields such as genetics [25] and astronomy [26]. Clustering techniques are clearly useful in any branch of science which concerns itself with ordering objects into synthetic groups.

Another class of models associated with unsupervised learning are generative models. These models aim to approximate some target distribution from which the training data is assumed to be sampled from. In Fig. 1.2, an example of both some training data which was used to train a generative model and a randomised selection of model generated data are shown for demonstrative purposes, the images were taken directly from [8].

In Chapter Two we will specifically look at energy based models, which have a clear analogy to models in statistical mechanics, with the specific goal of

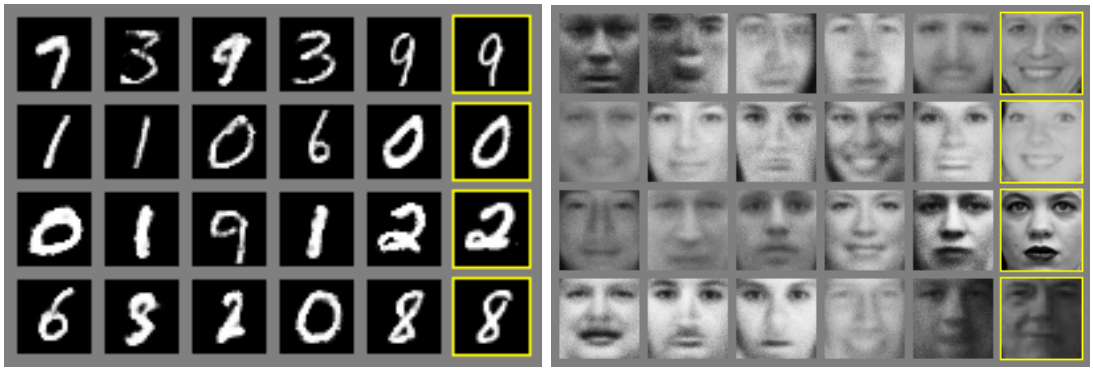

Figure 1.2 *Samples from the models presented in [8] for two typical datasets used in the ML community. Each image is a sample from a trained generative model, in the rightmost column, the nearest training example of the neighboring sample is shown. The model which generated the right sample was trained on the MNIST dataset [9], an open source dataset which contains (binary) black and white images of handwritten digits. The model which produced the left sample was trained on the Toronto face dataset [10], grey scale images of human faces.*

learning more about the model by studying the well known Ising model. Some generative models have an explicit and tractable expression for the model likelihood [27–29], by tractable we mean that the computation of the likelihood does not scale exponentially with the system size. As a result, these models can be trained using maximum likelihood. An interesting phenomenon which is rising in popularity in certain areas of physics [1, 30, 31] is to combine these kinds of models with traditional MC routines [32]. The work in Chapter Three will focus on one of these studies, which presents a proof of principle method for proposing lattice configurations for the Metropolis Hastings algorithm as opposed to a more traditional proposal generation technique such as local updates. Our work expands the set of models used in the original work and also begins a more systematic study of how the technique scales computationally with the system size.

It's not necessary to have an explicit expression for the model distribution however, for example Generative Adversarial Networks [8] (GANs) use two models: the generator and the discriminator, these are typically neural networks. The generator produces states which aim to be approximately distributed according to the distribution of the training data. The discriminator takes a state as input and tries to classify it as being genuine, taken from the training sample, or fake, generated from the model. The training process is iterative, keeping one model fixed and updating the parameters of the other and vice versa. GANs are notoriously difficult to train [33], but have achieved great success at generating sharp and

realistic looking images, see for example [34]. GANs are already being studied in physics, such as modelling particles in calorimeters [35, 36].

Something which isn't always entirely clear is what benefit generative models can provide. Since the models often require many training examples from the target distribution, the total amount of work required to train the model can prevent them from being efficient samplers in lieu of traditional MC techniques. Not least because the traditional MC techniques will likely still be required to produce a sizeable training sample. Furthermore, fully understanding all of the intricacies of sampling from a generative model is difficult at best, and in the worst case the model is used as a black box and therefore adds a layer of obscurity to the sampling process. A large amount of the work presented in this thesis is understanding exactly how the ML models are working. In Chapter Two the Restricted Boltzmann Machine requires a large number of states in order to be trained to a reasonable standard, however because of the model parametrization it's possible to learn about the physics of the system from the model using the expressions derived as part of that work. In Chapter Three we look at a model which falls into the class of auto-regressive models. These ostensibly seem like very attractive prospective models for areas of physics where there is a closed form expression for the distribution we wish to sample from, but in practice sampling from these distributions is highly non-trivial. Since these models don't require any pre-existing data in order to be trained to approximate a known distribution, they can potentially be used in scenarios where traditional algorithms struggle to sample the target distribution, for example systems with critical slowing down [37].

Regardless of the specific task, a difficulty which often arises with unsupervised learning is assessing the model performance. For generative models, one can show that the model distribution can reproduce the same moments or observables as the target distribution but this doesn't give a complete picture for how well the target distribution is reproduced. It also has the requirement that the true observable value can either be obtained analytically or from another sampling technique with high enough precision to make the comparison. There are potentially methods to assess model performance which bypass this, such as passing the states produced by the model through Metropolis-Hastings algorithm [38, 39] and measuring the acceptance, this approach was taken in [40] and will also be used to compare models in Chapter Three.

1.5 Bootstrap sampling estimators

In the later parts of this thesis there is abundant usage of bootstrap sampling in order to quantify the uncertainty on statistical estimators or observables. In many of these cases, we will often compare the same observable produced using different methods or models and in order to understand if the quantities agree in a statistical sense, we want to make some estimation of the precision to which they are determined. When there is a non-trivial dependence of the observable in question on the ensemble from which it was estimated, bootstrap sampling can be used to estimate the uncertainty [41, 42].

Here we will outline the concept of the bootstrap sample, which will serve as a reference for future chapters which rely on it. Consider a quantity $\mathcal{O}$ which is determined from a random sample $\boldsymbol{x} = x_1, x_2, \ldots, x_N$

$$\mathcal{O} = s(\boldsymbol{x}) \ , \tag{1.11}$$

where s is some function. A bootstrap sample of $\boldsymbol{x}$ is defined as

$$\boldsymbol{x}^b = x_1^b, x_2^b, \ldots, x_N^b \ , \tag{1.12}$$

where we have introduced a bootstrap sample index b. Note that $\boldsymbol{x}^b$ is not actually the dataset $\boldsymbol{x}$, but instead a *resampled* version of $\boldsymbol{x}$ where the sample $x_1^b, x_2^b, \ldots, x_N^b$ is sampled uniformly from $x_1, x_2, \ldots, x_N$ with replacement, E.g. you might have $x_1^b = x_4$, $x_2^b = x_N$, $x_3^b = x_4$, $x_3^b = x_1$ etc. Then, you can calculate the quantity of interest

$$\mathcal{O}^b = s(\boldsymbol{x^b}) \ , \tag{1.13}$$

where the index on the quantity indicates it was estimated from bootstrap sample b. Now you can estimate the standard deviation on $\mathcal{O}^b$ by repeatedly resampling $\boldsymbol{x}$ and taking the standard deviation of $\mathcal{O}$ across bootstrap samples

$$\mathrm{std}(\mathcal{O}) = \sqrt{\frac{1}{N_b - 1} \sum_{b=1}^{N_b} (\mathcal{O}^b - \bar{\mathcal{O}})} \tag{1.14}$$

where $\bar{\mathcal{O}}$ is the mean across bootstrap samples of $\mathcal{O}^b$. The beauty of using bootstrap sampling to estimate the uncertainty is that it requires no theoretical understanding of how the uncertainty propagates from $\boldsymbol{x}$ to $\mathcal{O}$. Furthermore, for many quantities the bootstrap mean and standard deviation can be computed

quickly.

Chapter 2

Restricted Boltzmann Machine

2.1 Introduction

A structured probabilistic model is a formalism used in machine learning, in particular deep learning, to describe the joint probability distribution of a set of random variables of interest and their interaction, via mathematical graphs. These models are often referred to as graphical models and consist of a set of vertices, connected by a set of edges. Over the years, various graphical models, together with their training and inference algorithms, have been developed by the machine learning community [43]. *Undirected* models are a popular subset of graphical models, often referred to as Markov Random Fields (MRFs), for which there is no directionality to the edges of the graph connecting the nodes [44]. A class of undirected models, with probability distribution of the form

$$p(\mathbf{v}) = \frac{e^{-E(\mathbf{v})}}{Z} \ , \tag{2.1}$$

are referred to as energy-based models (EBMs), or Boltzmann machines (BMs) if the models contains latent (hidden) variables [45–47]. In the above equation, $E(\mathbf{v})$ is the positive energy function and Z denotes the partition function. In this work, we focus on Restricted Boltzmann Machines (RBMs), a subset of BMs where the variables *within* the visible and hidden nodes are taken to be independent of each other [48]. RBMs are amongst the most common building blocks of deep probabilistic models. Our aim here is two-fold: the first is to be able to describe criteria that guarantee the machine is being adequately trained, as well as testing

its limitations; the second is to derive predictions from the RBM that cannot be directly obtained from the data.

The explicit form of the RBMs, together with their properties and training algorithms, are presented in Sec. 2.2. The RBM is trained for the 1- and 2-dimensional Ising models with volumes $L = 6$, $L^2 = 8 \times 8$ and $L^2 = 16 \times 16$, at various values of temperature. The Ising model and its properties are a paradigm in statistical physics literature, making it a suitable system on which to examine the training procedure and limitations of the RBMs [49–52]. Moreover, it is possible to generate a large number of Ising configurations using simple Monte Carlo simulations, avoiding the problem of small training sets.

We dedicate Sec. 2.3 and Sec. 2.4 to the details of our training procedure and the extraction of observables from the states generated by the trained RBM, such as magnetisation, energy, susceptibility and heat capacity. In particular, in Sec. 2.4, we recommend a prescription to ensure the machine is being trained correctly, by monitoring certain quantities during training. These quantities include measurements of the log-likelihood and the loss function, as well as the first and second moments of the distribution generated by the RBM. Following our training procedure, these observables are then shown to agree well with the expected results, obtained directly from the training data. Finally, we present a method for extracting the couplings between the spins of the Ising system, based on an observation in Ref. [53]. Using this method, RBMs not only give us a model for describing the distribution of the data, but also provide us with the predictive power of estimating the relative strength of the couplings between the visible nodes. Notice that due to the indirect all-to-all connections between visible nodes of an RBM, there are no pair-wise coupling assumptions in measuring the strength of connection between the nodes.

2.2 Training the RBM & making predictions

2.2.1 Restricted Boltzmann Machines

A Restricted Boltzmann Machine (RBM) is a type of undirected Markov Random Field (MRF) with a two layer architecture. The detailed derivation of some of the main features of the RBMs is clearly presented in Ref. [11], and we follow their

notation. An RBM consists of m visible nodes v_j, $j \in \{1, \cdots, m\}$, collectively denoted by $\mathbf{v}$ and representing the observed input data, and n hidden nodes h_i, $i \in \{1, \cdots, n\}$, collectively denoted by $\mathbf{h}$. In this work, we will focus on RBMs with binary variables, *i.e.* $v_j, h_i \in \{0, 1\}$. The energy of the joint state $\{\mathbf{v}, \mathbf{h}\}$ of the machine is as follows:

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i=1}^{n}\sum_{j=1}^{m} h_i w_{ij} v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i \,, \tag{2.2}$$

and we collectively call $\theta = \{\mathbf{w}, \mathbf{b}, \mathbf{c}\}$ the model parameters. The matrix $\mathbf{w}$ represents the undirected interaction between the visible and the hidden layers; with a slight abuse of language, in what follows a non-vanishing interaction between nodes will often be called a *connection*. The vectors $\mathbf{b}, \mathbf{c}$ are the biases of the visible and hidden layers respectively. Nodes on the same layer are not connected (making the machine *restricted*), leaving the interaction between them to be mediated by the connections to the other layer.

The RBM is used to encode the joint conditional probability distribution of a state $\{\mathbf{v}, \mathbf{h}\}$, given a set of parameters θ:

$$P(\mathbf{v}, \mathbf{h}|\theta) = \frac{e^{-E(\mathbf{v},\mathbf{h};\theta)}}{\mathcal{Z}(\theta)} \,, \tag{2.3}$$

where the partition function $\mathcal{Z}(\theta)$ normalizes the probability distribution and contains the sums over all possible states,

$$\mathcal{Z}(\theta) = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\theta)} \,. \tag{2.4}$$

The probability distribution of the variables in the visible layer is obtained by marginalising over the binary hidden variables h_i (see e.g. [11])

$$\begin{aligned} p(\mathbf{v}|\theta) &= \sum_{\mathbf{h}} P(\mathbf{v}, \mathbf{h}|\theta) = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\theta)} \\ &= \frac{1}{\mathcal{Z}(\theta)} \prod_{j=1}^{m} \left(e^{b_j v_j}\right) \prod_{i=1}^{n} \left(1 + e^{c_i + \sum_{j=1}^{m} w_{ij} v_j}\right) \,. \end{aligned} \tag{2.5}$$

The conditional probability of $\mathbf{v}$, given $\mathbf{h}$ and θ, and the one of $\mathbf{h}$, given $\mathbf{v}$ and θ, will be needed for training the RBM, and for generating configurations. Because the nodes of a given layer of an RBM are not connected these conditional

probabilities factorise

$$p(\mathbf{h}|\mathbf{v},\theta) = \prod_{i=1}^{n} p(h_i|\mathbf{v},\theta) \quad \text{and} \quad p(\mathbf{v}|\mathbf{h},\theta) = \prod_{j=1}^{m} p(v_j|\mathbf{h},\theta) \ , \tag{2.6}$$

and can be readily computed:

$$p(h_i = 1|\mathbf{v};\theta) = \sigma\left(\sum_{j=1}^{m} w_{ij}v_j + c_i\right) \ , \tag{2.7}$$

and

$$p(v_j = 1|\mathbf{h},\theta) = \sigma\left(\sum_{i=1}^{n} w_{ij}h_i + b_j\right) \ , \tag{2.8}$$

where $\sigma(x) = 1/(1+e^{-x})$ is the logistic function which, as was mentioned in Chapter 1, is often used as an activation function for neural network layers where the output has a probabilistic interpretation. Interestingly here, the sigmoid emerges naturally from marginalising over the visible and hidden layer respectfully.

In order to take advantage of a SGD based algorithm, the training dataset is partitioned into minibatches, *i.e.* each minibatch B contains a subset of the training data, $B = \{\mathbf{v}_1, ..., \mathbf{v}_l\}$. The RBM is trained by maximising the log-likelihood function on a given batch B with respect to the parameters $\theta = \{w_{ij}, b_j, c_i\}$

$$\log \mathcal{L}(\theta|B) = \log \prod_{v\in B} p(\mathbf{v}|\theta) = \sum_{v\in B} \log p(\mathbf{v}|\theta) \, . \tag{2.9}$$

Given an input vector $\mathbf{v}$,

$$\begin{aligned} \frac{\partial \log \mathcal{L}(\theta|\mathbf{v})}{\partial\theta} &= \frac{\partial \log p(\mathbf{v}|\theta)}{\partial\theta} \\ &= -\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v};\theta)\frac{\partial E(\mathbf{v},\mathbf{h};\theta)}{\partial\theta} + \sum_{\mathbf{v}',\mathbf{h}} P(\mathbf{v}',\mathbf{h}|\theta)\frac{\partial E(\mathbf{v}',\mathbf{h};\theta)}{\partial\theta} \, . \end{aligned} \tag{2.10}$$

More explicitly, the gradient with respect to the weights w_{ij} takes the form:

$$\frac{\partial \log \mathcal{L}(\theta|\mathbf{v})}{\partial w_{ij}} = p(h_i = 1|\mathbf{v};\theta)v_j - \sum_{\mathbf{v}'} p(\mathbf{v}'|\theta)p(h_i = 1|\mathbf{v}';\theta)v'_j \tag{2.11}$$

$$= p(h_i = 1|\mathbf{v};\theta)v_j - \mathbb{E}_{\text{model}}\left[p(h_i = 1|\mathbf{v}';\theta)v'_j\right] \ , \tag{2.12}$$

denoting average over the probability distribution of the model, $p(\mathbf{v}'|\theta)$, as

$$\mathbb{E}_{\text{model}}\left[F(\mathbf{v}')\right] = \sum_{\mathbf{v}'} p(\mathbf{v}'|\theta)F(\mathbf{v}')\,. \tag{2.13}$$

We can then average the gradients over a given batch, so that

$$\frac{1}{|B|}\sum_{\mathbf{v}\in B}\frac{\partial \log p(\mathbf{v}|\theta)}{\partial w_{ij}} = \mathbb{E}_{\text{data}}\left[p(h_i=1|\mathbf{v};\theta)v_j\right] - \mathbb{E}_{\text{model}}\left[p(h_i=1|\mathbf{v}';\theta)v'_j\right]\,, \tag{2.14}$$

where we have introduced the average over the dataset,

$$\mathbb{E}_{\text{data}}\left[F(\mathbf{v})\right] = \frac{1}{|B|}\sum_{\mathbf{v}\in B} F(\mathbf{v})\,. \tag{2.15}$$

Similarly, for the bias parameters,

$$\frac{1}{|B|}\sum_{\mathbf{v}\in B}\frac{\partial \log p(\mathbf{v}|\theta)}{\partial b_j} = \frac{1}{|B|}\sum_{\mathbf{v}\in B} v_j - \sum_{\mathbf{v}'} p(\mathbf{v}'|\theta)\mathbf{v}' \tag{2.16}$$

$$= \mathbb{E}_{\text{data}}\left[v_j\right] - \mathbb{E}_{\text{model}}\left[v'_j\right]\,, \tag{2.17}$$

$$\frac{1}{|B|}\sum_{\mathbf{v}\in B}\frac{\partial \log p(\mathbf{v};\theta)}{\partial c_i} = \frac{1}{|B|}\sum_{\mathbf{v}\in B} p(h_i=1|\mathbf{v};\theta) - \sum_{\mathbf{v}'} p(\mathbf{v}')p(h_i=1|\mathbf{v}';\theta) \tag{2.18}$$

$$= \mathbb{E}_{\text{data}}\left[p(h_i=1|\mathbf{v};\theta)\right] - \mathbb{E}_{\text{model}}\left[p(h_i=1|\mathbf{v}';\theta)\right]\,. \tag{2.19}$$

As shown above, the second terms in Eqs. 2.14, 2.16 and 2.18, *i.e.* terms that involve a model average $\mathbb{E}_{\text{model}}[.]$, contain sums over all possible states of the RBM; they are independent of the choice of the batch, and are computationally challenging. In practice, an approximation to the gradients is necessary, here we use the Contrastive Divergence (CD) algorithm [54]. The algorithm starts by choosing a training vector as the initial state of the visible units. Then a new state of the hidden units is generated using Eq. 2.7. This is done by setting a hidden unit h_i to be equal to 1 if the probability in Eq. 2.7 is greater than a random number uniformly distributed between 0 and 1; otherwise h_i is set to zero. Once the hidden units are chosen, a state of visible units, $\mathbf{v}^{(1)}$, is sampled according to the probability distribution Eq. 2.8, in an analogous procedure to sampling the hidden layer from the visible. These steps can be iterated several times, the number of iterations is controlled by a parameter, which we denote by k in what follows. Finally, the gradient is computed using Eq. 2.14, where $\mathbf{v}' = \mathbf{v}^{(k)}$. We refer to this procedure of generating a state from the model as the

CD loop [55], and refer to k as the number of CD steps. Having split the training dataset into minibatches, and calculated an expression for the gradient of the likelihood with respect to the parameters we have the necessary ingredients to optimise the parameters using any form of stochastic gradient descent. In this work the parameters θ are updated using the basic SGD algorithm, outlined in Chapter 1, by just taking a step of size α, the *learning rate*, in the direction of the gradient.

A cycle over all training batches, an *epoch*, results in a number of parameter updates equal to the number of batches. It is important for the purposes of our study to be able to quantify the quality of the training procedure. Therefore, at each epoch during the training procedure, we monitor several indicators:

1. The maximisation of the average log-likelihood. The average is computed first within each batch and then averaged over all batches. The computation of the log-likelihood requires the intractable partition function to be estimated. This estimate has been done using the annealed importance sampling procedure, discussed in detail in Sec. 2.2.2.

2. For a given state $\mathbf{v}$, $F(\mathbf{v};\theta)$ is defined as

$$F(\mathbf{v};\theta) = \log \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\theta)} \,. \tag{2.20}$$

   This quantity is averaged first across the states in each batch and then across all the batches, and it is expected to reach a constant once the machine is trained. To see this more clearly, notice that we can write the log-likelihood in Eq. 2.9 as:

$$\sum_{v\in B} \log p(\mathbf{v}|\theta) = \sum_{v\in B} F(\mathbf{v};\theta) - \log(Z) \,. \tag{2.21}$$

   Since the log-likelihood is a quantity we are trying to maximise, once the machine is trained, it converges to a constant. Similarly for the estimation of the partition function $\log(Z)$. Therefore, during the training procedure, we can monitor how $F(\mathbf{v};\theta)$ is also progressing to a constant.

3. The loss function is defined as the average over the batches of the quantity

$F\left(\mathbf{v}^{(0)};\theta\right) - F\left(\mathbf{v}^{(k)};\theta\right)$,

$$\text{Loss} = \frac{1}{N_{\text{batch}}} \sum_{\text{batches } b} \frac{1}{|B^{(b)}|} \sum_{v \in B^{(b)}} \left(F\left(\mathbf{v}^{(0)};\theta\right) - F\left(\mathbf{v}^{(k)};\theta\right)\right) \tag{2.22}$$

where $\mathbf{v}^{(0)}$ is the input data from the training set, $\mathbf{v}^{(k)}$ is the state reconstructed by the machine using k steps in the CD loop, and N_{batch} is the total number of batches.

4. The reconstruction error ϵ, is defined as

$$\epsilon = \frac{1}{N_{\text{batch}}} \sum_{\text{batches } b} \frac{1}{|B^{(b)}|} \sum_{v \in B^{(b)}} |\mathbf{v}^{(0)} - \mathbf{v}^{(1)}|^2 \tag{2.23}$$

where $\mathbf{v}^{(0)}$ is as stated above and $\mathbf{v}^{(1)}$ is the state reconstructed by the machine using a single step of Gibbs sampling.

In summary, if the training procedure is correct, the log-likelihood should be an increasing function of the training steps, approaching a plateau where it stabilises at a fixed value. The quantity $F(\mathbf{v};\theta)$ is expected to reach a constant value as described above. The loss function and the reconstruction error are expected to decrease along the training, as the machine is required to reduce the difference between the generated $\mathbf{v}$'s and those used for training.

2.2.2 Annealed importance sampling

As discussed in the previous section, training the RBM corresponds to maximising the log-likelihood on the input training set, with respect to parameters of the model. This in turns requires the partition function for the model to be computed, which is intractable in most cases; the computation scales exponentially with the number of visible and hidden units in the machine. However, there are methods to estimate the partition function. A simple method is to use a proposal distribution $p_0(\mathbf{v}) = \frac{1}{Z_0} p_0^*(\mathbf{v})$ whose partition function is tractable and can be measured exactly. The asterisk here indicates that the probability is unnormalized, with its corresponding partition function being the normalization factor. Suppose we wish to estimate an intractable partition function Z_1 for $p_1(\mathbf{v}) = p_1^*(\mathbf{v})/Z_1$.

We can write

$$Z_1 = \int p_1^*(\mathbf{v}) d\mathbf{v} = Z_0 \int p_0(\mathbf{v}) \frac{p_1^*(\mathbf{v})}{p_0^*(\mathbf{v})} d\mathbf{v} \ , \tag{2.24}$$

and measure its estimator

$$\hat{Z}_1 = \frac{Z_0}{M} \sum_{m=1}^{M} \frac{p_1^*(\mathbf{v}^{(m)})}{p_0^*(\mathbf{v}^{(m)})} \, , \tag{2.25}$$

where the sample $\{\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(M)}\}$ is drawn according to $p_0(\mathbf{v})$. One can readily observe that if the two distributions p_0 and p_1 have large overlap, states that are drawn with a high probability from p_0, also have a high probability in p_1, and the sum is close to the original integral. In contrast, if the overlap is poor, the drawn states make negligible contribution to the sum, making it a biased estimator for $\hat{Z}_1$. The issue can be further quantified by computing the variance of $\hat{Z}_1$:

$$\mathrm{Var}[\hat{Z}_1] = \frac{Z_0^2}{M^2} \mathrm{Var} \left[\sum_{m=1}^{M} \frac{p_1^*(\mathbf{v}^{(m)})}{p_0^*(\mathbf{v}^{(m)})} \right] = \frac{Z_0^2}{M^2} \sum_{m=1}^{M} \mathrm{Var} \left[\frac{p_1^*(\mathbf{v}^{(m)})}{p_0^*(\mathbf{v}^{(m)})} \right] = \frac{Z_0^2}{M} \mathrm{Var} \left[\frac{p_1^*(\mathbf{v})}{p_0^*(\mathbf{v})} \right] , \tag{2.26}$$

where in the second equality, it is assumed that the samples are drawn independently. Expanding the variance of the ratio, we obtain

$$\mathrm{Var} \left[\frac{p_1^*(\mathbf{v})}{p_0^*(\mathbf{v})} \right] = \int p_0(\mathbf{v}) \left[\frac{p_1^*(\mathbf{v})}{p_0^*(\mathbf{v})} - \frac{Z_1}{Z_0} \right]^2 d\mathbf{v} = \frac{Z_1^2}{Z_0^2} \int p_0(\mathbf{v}) \left[\frac{p_1(\mathbf{v})}{p_0(\mathbf{v})} - 1 \right]^2 d\mathbf{v} \, . \tag{2.27}$$

Hence, the estimate for the variance of $\hat{Z}_1$ becomes

$$\hat{\mathrm{Var}}[\hat{Z}_1] = \frac{\hat{Z}_1^2}{M^2} \sum_{m=1}^{M} \left[\frac{p_1(\mathbf{v}^{(m)})}{p_0(\mathbf{v}^{(m)})} - 1 \right]^2 \, . \tag{2.28}$$

If the two distributions p_0 and p_1 are not close to each other, in the Monte Carlo simulation, there will be many samples drawn from p_0 with a high probability which have a corresponding low probability with respect to p_1, leading to a large contribution to the sum in Eq. 2.28. In fact, it has been discussed in Refs. [56, 57] that the above variance can become large and even infinite, leading to very poor estimates.

In most cases, as it is with the RBM, p_1 is a multimodal distribution over a high

dimensional space. As a result, it is difficult to propose a simple distribution p_0 with a tractable partition function that gives a good estimate for the original Z_1. In such situations a method known as annealed importance sampling is adopted [57–60]. This method tries to bridge the original distribution p_1 and the simple distribution p_0, by introducing intermediate closer distribution $p_{\beta_0}, p_{\beta_1}, \cdots, p_{\beta_n}$ such that $0 = \beta_0 < \beta_1 < \cdots < \beta_{n-1} < \beta_n = 1$. We therefore try to estimate Z_1/Z_0 via:

$$\frac{Z_1}{Z_0} = \frac{Z_{\beta_1}}{Z_0}\frac{Z_{\beta_2}}{Z_{\beta_1}} \cdots \frac{Z_{\beta_{n-2}}}{Z_{\beta_{n-1}}}\frac{Z_1}{Z_{\beta_{n-1}}} = \prod_{j=0}^{n-1} \frac{Z_{\beta_{j+1}}}{Z_{\beta_j}} \ , \tag{2.29}$$

where $\frac{Z_{\beta_{j+1}}}{Z_{\beta_j}}$ can be estimated using the simple importance sampling Eq. 2.25 described above. These estimates are reliable as the corresponding distributions are close. The intermediate distribution can be chosen such that they are proportional to the geometric average of p_1 and p_0, *i.e.* ,

$$p_\beta \propto p_1^*(\mathbf{v})^\beta p_0^*(\mathbf{v})^{1-\beta} \, . \tag{2.30}$$

To see why this is a sensible choice, notice that

$$p_1^*(\mathbf{v})^\beta p_0^*(\mathbf{v})^{1-\beta} = e^{-\beta E_1(\mathbf{v})} e^{-(1-\beta)E_0(\mathbf{v})} = e^{-E_0(\mathbf{v})} e^{-\beta[E_1(\mathbf{v}) - E_0(\mathbf{v})]} \, . \tag{2.31}$$

In other words, when β is very small, p_β is also very close to the simple distribution p_0. Increasing β gradually, until it is close to one, will have p_β being close to p_1, as required. Here, we make the choice $E_0 = 0$, *i.e.* , we sample from a uniform starting distribution $p_0^*(\mathbf{v})$ for $\beta_{j=0} = 0$. This implies that, for $j = 1, 2, \cdots, n$,

$$p_{\beta_j}^*(\mathbf{v}) \propto e^{-\beta_j E_1(\mathbf{v})} \, , \tag{2.32}$$

where $e^{-E_1}(\mathbf{v})$ is the numerator term in Eq. 2.5. In the first step $\mathbf{v}$ is sampled from a uniform distribution and $p_{\beta_1}^*$ is computed according to Eq. 2.32, for a value of β close to zero. A set of configurations $\mathbf{v}$ is sampled according to that probability using Gibbs sampling, after having taken into account the presence of β in Eqs. 2.7 and 2.8, where it enters as a multiplicative factor in the argument of the logistic functions. The number of constrastive divergence steps, as described in the previous section, used to generate the next set of configurations from the previous is taken to be $k_{\text{CD}} = 1$. We chose to increase β by 0.0001 in each step and continue until $\beta_n = 1$ is reached. The number of states, M, used to estimate $\frac{Z_{\beta_{j+1}}}{Z_{\beta_j}}$ was chosen to be 100. As a practical consideration, we are interesting in

calculating the log-likelihood and so we calculate the logarithm of the ratio of the partition functions, leading to a sum of logarithms of the bridging ratios. This has the added advantage of avoiding numerical overflow which would occur when taking the ratio of the partition functions.

2.2.3 1D and 2D Ising model simulations

We train the RBM on Ising spin configurations in 1D and 2D, distributed according to the Boltzmann weight:

$$P(s|J) = \frac{e^{-H_J(s)}}{Z_{\text{Ising}}} = \frac{e^{-J/(k_B T)\sum_{\langle i,j\rangle} s_i s_j}}{Z_{\text{Ising}}} \ , \tag{2.33}$$

where $\langle i, j\rangle$ indicates that the sum is over nearest neighbour spins, T is the temperature of the system and k_B is the Boltzmann constant. The partition function is the sum over all possible configurations of the system,

$$Z(J) = \sum_s e^{-H_J(s)}. \tag{2.34}$$

Each Ising spin can take a binary value -1 or 1, and the external field is set to zero. The 2D Ising configurations were generated using `Magneto` [61], a fast parallel C++ code available online. Magneto uses the Swendsen-Wang Monte Carlo algorithm [62] to generate the configurations minimising the autocorrelation. For simplicity we set the combination $J/k_B = 1$, and the temperature becomes the defining feature in differentiating the behaviour of various Ising systems in a given number of dimensions. Note that spins ± 1 are converted to $0, 1$ before setting them as input to the RBM.

The expectation value for an Ising observable at a given temperature,

$$\langle O\rangle\,|_T = \sum_{\mathbf{s}} p(\mathbf{s}|T) O(\mathbf{s})\,, \tag{2.35}$$

is estimated by the average of the measured observable over a finite set of independent configurations:

$$\langle \hat{O}\rangle|_T = \frac{1}{N}\sum_{n=1}^{N} O\left(\mathbf{s}(n), T\right) \ , \tag{2.36}$$

where N is the total number of sampled states and $\mathbf{s}^{(N)} \sim p(\mathbf{s}|T)$. The observables under considerations are the average magnetisation, susceptibility, energy and heat capacity:

$$\begin{aligned}
\langle m \rangle &= \frac{1}{L^2} \left\langle \left| \sum_{i=1}^{L^2} s_i \right| \right\rangle , \\
\langle \chi \rangle &= \frac{L^2}{T} \left\langle \langle m^2 \rangle - \langle m \rangle^2 \right\rangle , \\
\langle E \rangle &= -\frac{1}{L^2} \left\langle \sum_{\langle i,j \rangle} s_i s_j \right\rangle , \\
\langle c_v \rangle &= \frac{L^2}{T^2} \left\langle \langle E^2 \rangle - \langle E \rangle^2 \right\rangle .
\end{aligned} \tag{2.37}$$

Once the training process is completed we generate configurations according to the probability encoded by the RBM. We then compare the results with the corresponding values from Magneto. The methods for generating new configurations from the RBM distribution, as well as measurements of observables in the case of the 2-dimensional Ising model are given in detail in Sec. 2.4.

2.3 Validation in one dimension

To validate our code, we trained a machine on a set consisting of 1-dimensional Ising states with 6 sites, sampled from a distribution with $J = T = 1$, similar to Ref. [50]. The training was divided into three steps of 2000, 1000 and 1000 epochs respectively, with a decreasing learning rate at each step, as summarised in Table 2.1. In this case the analytical form of the probability distribution is relatively easy to compute so it was used to validate our implementation of both the training code and the annealing algorithm, similar to Ref. [50].

The results for this test are shown in Fig. 2.1. For a 1D chain with 6 sites, there are 64 different possible states. The x-axis of the plot on the top plot of Fig. 2.1 indexes these states, with their corresponding probabilities on the y-axis. The true probability of these states, measured exactly, has been plotted with a black dashed line. The prediction for the underlying distribution improves as the number of training epochs increases, with the colour blue corresponding to the learned distribution at 20 epochs and the colour red indicating the learned distribution at epoch 4000, which is very close to the true distribution. The bottom plot of

Training phase	Epochs	LR α	k_{CD}
1	2000	0.01	1
2	1000	0.001	1
3	1000	0.0001	1

Table 2.1 *Parameters for training an RBM on the 1-dimensional Ising model with six spins. The model has six visible nodes (fixed by the number of spins in the training examples) and six hidden nodes. The training dataset consisted of 100000 spin configurations, split into batches of size 200. There are a total of 64 possibles states, making it possible for the partition function to be measured exactly. We start the training with $\alpha = 0.01$ and gradually fine-tune it to smaller values as the training progresses. In this cases keeping $k = 1$ throughout the training is sufficient for learning the distribution of the data accurately.*

Fig. 2.1 shows, in blue, how the true log-likelihood increases vs epochs, while the annealing algorithm prediction, plotted in orange, is able to reproduce the exact log-likelihood with approximately 10% error, keeping the same overall functional form. We will assume that this is the typical order of magnitude for accuracy of the log-likelihood resulting from the annealing algorithm; and will use the log-likelihood computed with the annealing procedure as our measure for the effectiveness of a training procedure, particularly for more complicated systems where measurement of the true partition function is intractable. Notice that the annealing underestimates Z, thus resulting in an upper bound for the true log-likelihood [57].

At the end of the training we expect

$$\frac{e^{-H(v)}}{Z^{\mathrm{Ising}}} = \frac{e^{-H_\lambda^{\mathrm{RBM}}(v)}}{\mathcal{Z}}, \tag{2.38}$$

which, for every v, implies

$$H(v) - H_\lambda^{\mathrm{RBM}}(v) = \log \mathcal{Z} - \log Z^{\mathrm{Ising}} = \text{constant}. \tag{2.39}$$

We can reformulate this statement in the following way: minimising the KL divergence only requires the difference between the Ising probability distribution

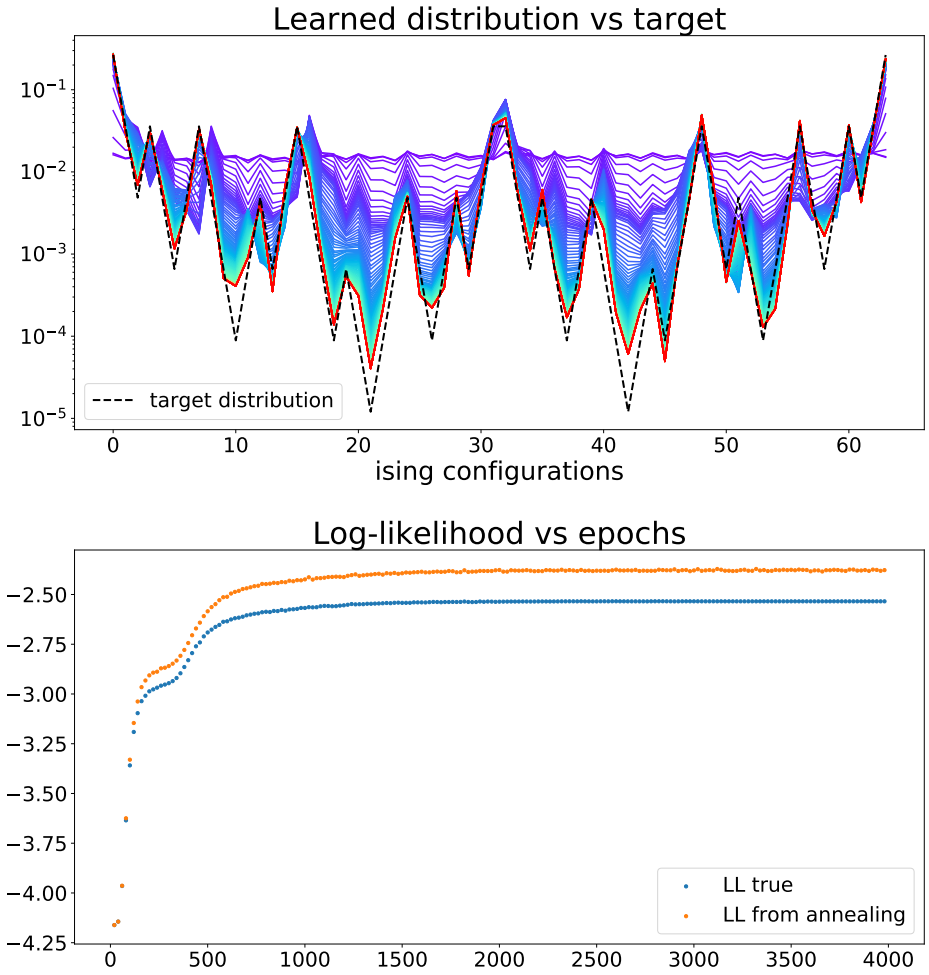

Figure 2.1 *The top plot shows the RBM distributions obtained at different number of training epochs, from 20 (blue) to 4000 (red) compared to the target distribution (dotted black line). The bottom plot presents is a comparison between the exact log-likelihood (blue) and that obtained from the annealing algorithm (orange). The uncertainty for the estimated log-likelihood is smaller than* 10%.

and that of the RBM to go to zero

$$D_{\mathrm{KL}}\Big[q_{\mathrm{Ising}}(\mathbf{v})||p_\theta(\mathbf{v})\Big] = \sum_{\mathbf{v}} \Big(q_{\mathrm{Ising}}(\mathbf{v}) \log\left(q_{\mathrm{Ising}}(\mathbf{v})\right) - q_{\mathrm{Ising}}(\mathbf{v}) \log\left(p_\theta(\mathbf{v})\right)\Big)$$
$$= \sum_{\mathbf{v}} q_{\mathrm{Ising}}(\mathbf{v}) \Big[(\log \mathcal{Z} - \log Z^{\mathrm{Ising}}) - (H(\mathbf{v}) - H_\lambda^{\mathrm{RBM}}(\mathbf{v}))\Big], \tag{2.40}$$

which in turn implies that the expectation value of the Hamiltonians over the probability distribution of the Ising model may differ by a constant, which is

cancelled in the equation above. Note that the statement should also be true term-by-term once the machine has learned, *i.e.* for any state $\{\mathbf{v}\}$. This is observed numerically, where

$$H(\mathbf{v}) - H^{\mathrm{RBM}}(\mathbf{v}) = \log \mathcal{Z} - \log Z^{\mathrm{Ising}} \ , \ \forall \, \mathbf{v} \tag{2.41}$$

up to numerical errors, for all the 64 possible states.

In order to verify if this is indeed the case in the training the machine, in Fig. 2.2, we plot the average $\langle H - H^{\mathrm{RBM}} \rangle_v$ of the 64 energy differences in Eq. 2.41, together with their corresponding standard deviation $\sigma(H - H^{\mathrm{RBM}})_v$, as a function of the training epochs. Notice that both the average and the standard deviation converge to constant values, -10.56 and 0.55 respectively, showing how the differences of Eq. 2.41 indeed converge to a common constant value of -10.56 up to a numerical error of 0.55. We have also checked that these numerical values are consistent with $\log \mathcal{Z} - \log Z^{\mathrm{Ising}}$, which is plotted, in orange, together with $\langle H - H^{\mathrm{RBM}} \rangle_v$ in Fig. 2.2: the former lies within 1 sigma from the latter, showing how the KL divergence in Eq. 2.40 indeed decreases towards zero along the training.

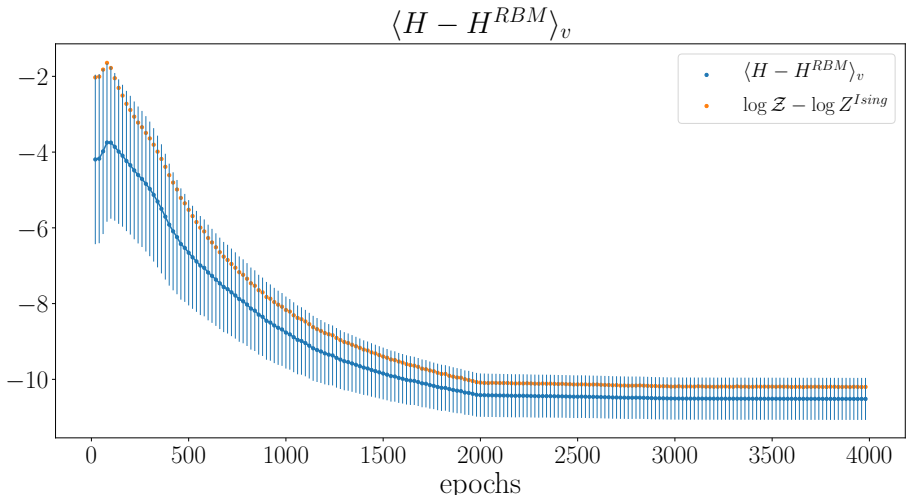

Figure 2.2 *We plot* $\langle H - H^{RBM} \rangle_v$ *(blue) and* $\log \mathcal{Z} - \log Z^{Ising}$ *(orange) per epoch. The vertical lines indicate standard deviation of* $\langle H - H^{RBM} \rangle_v$.

2.4 RBM for the 2D Ising model

We now consider training sets consisting of spin configurations of 2-dimensional Ising systems at various temperatures. The configurations are generated using

Magneto, setting $J/k_B = 1$, with temperatures above, at and below the critical temperature, starting from $T = 1.8$ to $T = 3.0$ in steps of $\Delta T = 0.1$. The training of each RBM on these configurations has been studied independently, with the aim of obtaining the set of parameters which provide the correct value of observables at each temperature, as compared with Magneto values. The observables for each trained RBM have been computed as described in Sec. 2.2.3, for different checkpoints along the training, and plotted as a function of the number of epochs. This allows us to monitor the training procedure, and to assess the number of epochs required to converge to the underlying distribution of the training data. This procedure has been performed for the case of an 8^2 lattice, as well as a 16^2 lattice. We highlight the increasing difficulties in training when the volume is increased.

2.4.1 Ising simulation and RBM training parameters

The simulation parameters for generating Ising spins using Magneto [61] and training the RBM are presented in Tables 2.2 and 2.3 respectively. In the following we present detailed results for three systems of 8^2 spin configurations, at temperatures $T = 1.8$, $T = 2.2$ and $T = 3.0$ respectively. The procedure for training a system of 16^2 spins is reported in App. A.2.

L^2	N_{therm}	N_{measure}	Binning	algorithm	T	Steps ΔT
8^2	50000	100000	1	SW	1.8-3.0 & 2.27	0.1
16^2	50000	100000	1	SW	1.8-3.0 & 2.27	0.1

Table 2.2 *2D Ising model parameters for data generation. Using the Swendsen-Wang (SW) algorithm, the autocorrelation drops below 1 therefore no binning was required. N_{therm} denotes the number of MC iterations used for thermalisation, while $N_{measure}$ denotes the measurement taken after thermalisation, which are saved to be used as training examples for the RBM.*

In order to train the RBMs we used 10^5 Ising configurations split into minibatches with size 200. Although the training procedure is specific to each dataset and RBM architecture, we give a general prescription for the case where the number of hidden and visible nodes are chosen to be the same. Most our trainings were performed in three steps:

1. We start the training with a large learning rate α, the value of which has

to be tested for. The idea for such a choice is to speed up the process of maximising the log-likelihood at the beginning. The number of CD steps k_{CD}, at this stage, is set to 1. The first training phase was ran until the log-likelihood plateaus. In some cases if the first training phase is ran for long enough, the log-likelihood starts to decrease, an example of this behaviour is reported in the Appendix in the case of $T = 1.8$. This behaviour is observed elsewhere in the literature, *e.g.* see Ref. [11].

2. In order to prevent the decrease in the log-likelihood, we decrease the variance in the estimate of the gradient of the log-likelihood, namely we increase the CD parameter to a larger value. Again, the value is chosen according to whether or not it results in an increase in the log-likelihood. At this stage, we also decrease the learning rate α, in order to fine-tune the training as we approach the maximum of the log-likelihood.

3. A single fine-tuning such as the one described above may not be enough to fully train the machine. In that case, we iterate step 2 with a higher value of k_{CD} and lower value of α. This has been done for all our trained machines.

4. At each of the above steps, we monitor the moments of the learned distribution and compare the values to the expected ones, obtained directly from the training set. This allows us to monitor if and how the training is progressing.

Therefore, the training of each machine in our case is characterised by three different phases with specific number of epochs, learning rates, and values of k, as well as comparing the moments extracted from the machine with the expected values obtained from the training data. Our choices for the hyperparameters are summarised in Table 2.3.

The four quantities used to monitor training, *i.e.* the log-likelihood, free energy, loss function and reconstruction error are reported as functions of the training epoch in Fig. 2.3, for three different temperatures. They all have the expected behaviour, as described in Sec. 2.2.1. We stop the training process when there is no further increase in the log-likelihood and, after fine-tuning with a higher value of k and smaller α, the first and second moments of distributions generated by machines at different epochs are statistically equivalent. We can conclude that the training algorithm, so far, has been successful. In the next section we assess how well the machine is able to predict the observables, as compared to the measured Magneto values.

Visible	Hidden	T	ΔT	Training Phase	Epochs	LR α	k_{CD}
8^2	8^2	1.8-2.1	0.1	1	3000	0.1	1
				2	1000	0.01	5
				3	1000	0.001	10
8^2	8^2	2.2-3.0	0.1	1	2000	0.01	1
				2	1000	0.001	5
				3	1000	0.0001	5
16^2	16^2	1.8	//	1	8000	0.01	10
				2	2000	0.01	20

Table 2.3 *Parameters for training the RBMs. The number of visible and hidden nodes for each case is presented. The dataset for each model comprised of 100000 training examples, split into minibatches of size 200. T indicates the temperatures of the given systems, with intervals* $\Delta T = 0.1$. *For each training phase we record the learning rate (LR),* α *and number of contrastive divergence steps,* k_{CD}. *As advised in Ref. [11], whenever the increase in the log-likelihood plateaus and then starts to decrease, we reduce the value of* α *and increase* k_{CD}. *The number of epochs in each phase depends on the size and temperature of each system. Our tests indicate that a higher value of* k, *i.e.* $k = 10$, *needs to be used to train the machine on the larger configuration (*16^2 *spins), which is then increased to* $k = 20$ *for the last phase of the training.*

2.4.2 Gibbs sampling

As anticipated in Sec. 2.2.3, for each trained machine we have computed the average magnetisation, susceptibility, energy and heat capacity, together with their corresponding uncertainties. The results have been compared to the values obtained from the training set, in order to examine if our machines are able to reproduce the first and second moments of the target distribution. In order to sample the states from the RBM distribution, required to implement Eq. (2.36), we have used Gibbs Sampling, Ref. [11]. Block Gibbs sampling is a particular Markov Chain Monte Carlo method (MCMC). The general idea behind any MCMC method is that of setting up a Markov Chain that converges to the distribution we want to sample from, in this case Eq. (2.3), and then running the chain long enough to reach its stationary point. In the case of Gibbs sampling, this translates into building the chain

$$\mathbf{v}^{(0)} \to \mathbf{h}^{(0)} \to \mathbf{v}^{(1)} \to \mathbf{h}^{(1)} \to \mathbf{v}^{(2)} \quad \cdots \ ,$$

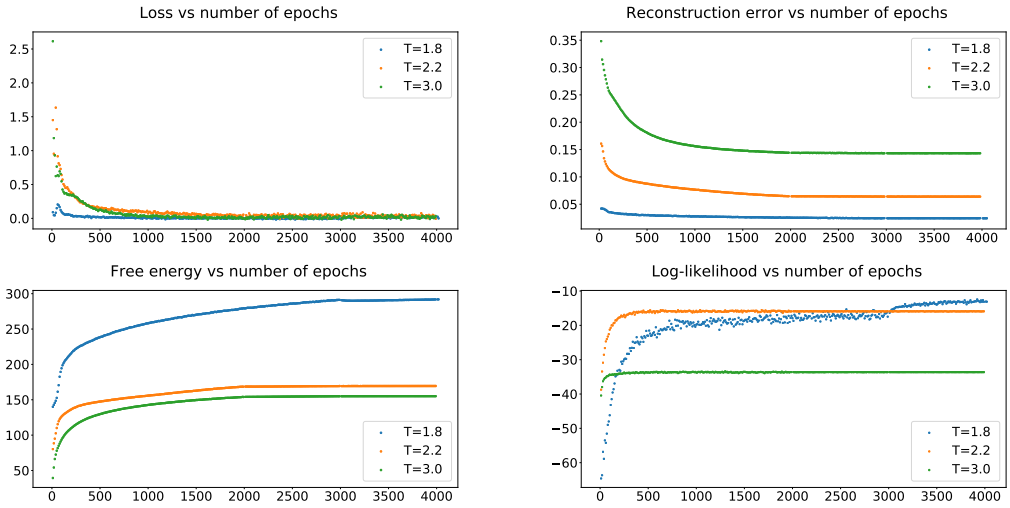

Figure 2.3 *From left to right, loss function and reconstruction error in the first line, free energy and log-likelihood in the second one. The results are presented for three different* $T = 1.8$*,* $T = 2.2$ *and* $T = 3.0$*.*

where $\mathbf{h}^{(l)}$ and $\mathbf{v}^{(l+1)}$ are sampled from the conditional probabilities given in Eq. 2.7 and Eq. 2.8 respectively. It can be shown that this chain converges to $P(\mathbf{v}, \mathbf{h}; \theta)$, which implies that at a certain point in the chain, the states $\mathbf{h}^{(l)}$ and $\mathbf{v}^{(l)}$ will be sampled according to the joint probability distribution encoded in the RBM. The word *block* here, refers to the fact that we can simultaneously update all variables h_i given $\mathbf{v}^{(l)}$, and vice versa, as the RBM is a restricted network which means that the variables within each layer are independent of each other. Note that the Contrastive Divergence algorithm, used to obtain estimates of the log-likelihood gradient as described in Sec.2.2, is Gibbs sampling run with k_{CD} Gibbs steps, to get $\mathbf{v}^{(k)}$ from $\mathbf{v}^{(0)}$ through $\mathbf{v}^{(0)} \to \mathbf{h}^{(0)} \to \ldots \to \mathbf{v}^{(k)}$. One of the main advantages of Gibbs sampling over other MCMC methods (see the next section on Metropolis sampling for an example) is that it converges relatively fast. This allowed us to measure and plot the observables at many epochs along the training, together with their corresponding uncertainties, as an indication of progress in learning in Fig. 2.4. The details concerning the Gibbs sampling parameters are reported in Table (2.4). After a thermalisation step of 2×10^4 configurations, we measure magnetisation and energy on every 100th configuration, allowing the Markov chain to run for 2×10^6 configurations.

Algorithm	N_{therm}	N_{measure}	Binning
Gibbs	20000	2×10^6	100
Metropolis	50000	1×10^6	50

Table 2.4 *Gibbs and Metropolis sampling parameters, for MC simulations on an already trained RBM. The measurements are made on configurations generated after the initial thermalisation step N_{therm}. The binning factor indicates the number of successive measurement binned to ensure the remaining are indeed independent.*

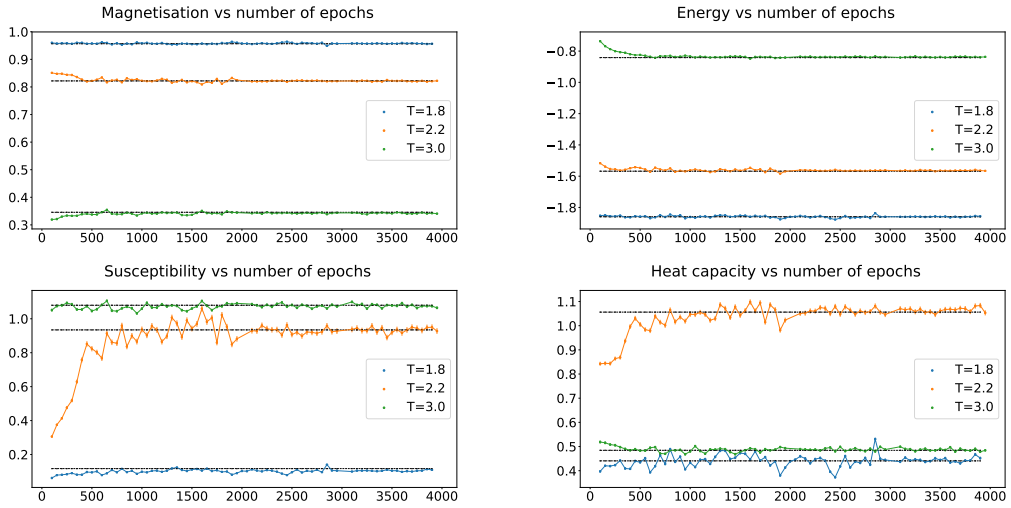

Figure 2.4 *From left to right, magnetization and energy in the first line, susceptibility and heat capacity in the second one. The results are presented for three different T=1.8, T=2.2 and T=3.0, as function of the training epoch.*

2.4.3 Metropolis sampling

In order to have an independent measurement of the observables, we implemented the Metropolis algorithm. The algorithm takes, as input, the parameters of a trained RBM, $\theta = \{\mathbf{w}, \mathbf{b}, \mathbf{c}\}$. Starting from a random configuration, a new spin configuration is proposed by randomly flipping a spin using the transformation $v_i = 1 - v_i$. Letting u to be a random number between zero and one from the uniform distribution, if,

$$\min\left(1, \frac{p_{\text{RBM}}(\mathbf{v}_{\text{new}})}{p_{\text{RBM}}(\mathbf{v}_{\text{old}})}\right) > u \ , \tag{2.42}$$

we accept the proposed configuration. Otherwise we reject the new configuration and return to the original one. At each Monte Carlo sweep, this process is

repeated for all spin variables. Since we take the ratio between the new and the old probabilities, the partition functions cancel and so it does not need to be estimated. The procedure is repeated until the thermalisation is reached. These can be checked by checking that the Monte Carlo history of an observable e.g. magnetisation, follows a normal distribution. We chose to discard the first 5×10^5 configurations for the thermalisation step. See *e.g.* Fig. A.15 in the Appendix. Note that the generated configurations in the $0, 1$ basis are mapped to those of the Ising model in the $-1, 1$ according to $v_{\text{ising}} = 2v_{RBM} - 1$.

In order to take independent measurements of the observables, successive number of measurements of $|m|$ and E were binned. To determine the bin size, the error on *e.g.* $|m|$ vs bin size was plotted in Fig. A.17. To confirm independence, the autocorrelation time defined as:

$$A(T) = \frac{1}{N_{\text{meas}} - t} \left[\sum_{i=1}^{N_{\text{meas}}-t} m^{(i)} m^{(i+t)} - (\overline{m}^{(i)})^2 \right] / \left(\overline{(m^{(i)})^2} - (\overline{m}^{(i)})^2 \right) \ , \quad (2.43)$$

where N is the total number of Monte Carlo steps, was also computed. The autocorrelation time vs MC steps is plotted in Fig. A.18. Based on these two plots, choosing bin size equal to 50, seems sufficient for independent measurements to be obtained. We present the parameters of the Metropolis sampling in Table. 2.4. The magnetisation $\langle |m| \rangle$, energy $\langle E \rangle$, susceptibility χ and heat capacity C_V, are then measured. The error on these quantities have been computed using the statistical bootstrap procedure.

We also tried to extract the third and fourth moments, however, the corresponding uncertainties turned out to be of the same order as the moments themselves.

2.4.4 Observable predictions at all temperatures

In the following we report on the results of Ising observable measurements, as extracted from a trained RBM, for 13 different values of temperature. In Fig. 2.5 the values for the magnetisation and susceptibility are shown, obtained independently from Gibbs and Metropolis sampling. The expected values from magneto are also plotted. In Fig. 2.6, the analagous plots are shown for energy and heat capacity.

We can observe resulting curves reproduce the expected values for magnetisation, energy, susceptibility and heat capacity.

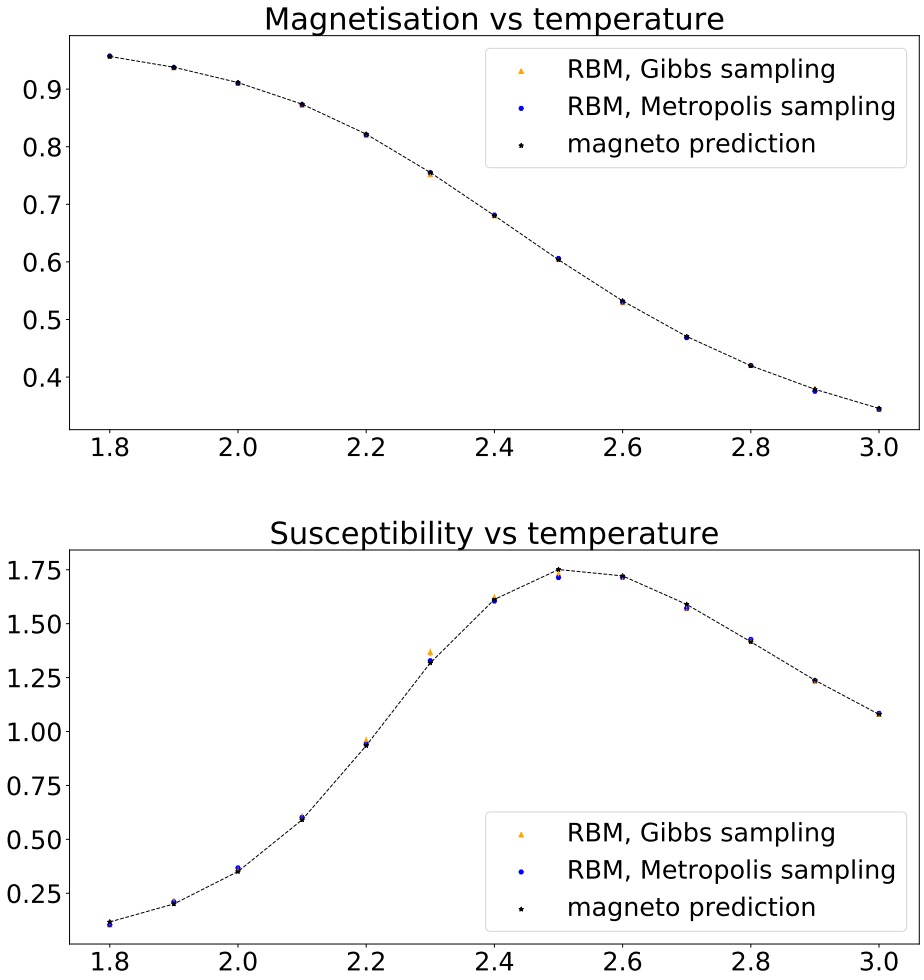

Figure 2.5 *Magnetisation and susceptibility vs temperature. Observables are estimated on samples from performing Gibbs and Metropolis sampling on the trained RBMs. The model values are compared to the expected values from Magneto.*

However in these plots it is not possible to discern between the results obtained by different sampling methods and expected magneto values. To indicate more clearly the precision with which the RBM is able to reproduce the moments, we plot the observables normalized over their expected values, with their corresponding error bars, for magnetisation and susceptibility in Fig. 2.7, and energy and heat capacity in Fig. 2.8. Notice that, as expected, the first moments *i.e.* magnetisation and energy, agree better with the corresponding magneto results as compared to the second moments. Moreover, the measurements using the two different sampling methods always agree within 2σ. For the first moments, the values measured from the RBM distribution are almost always compatible with the expected values within 2σ, while in the case of the second moments some discrepancy is observed,

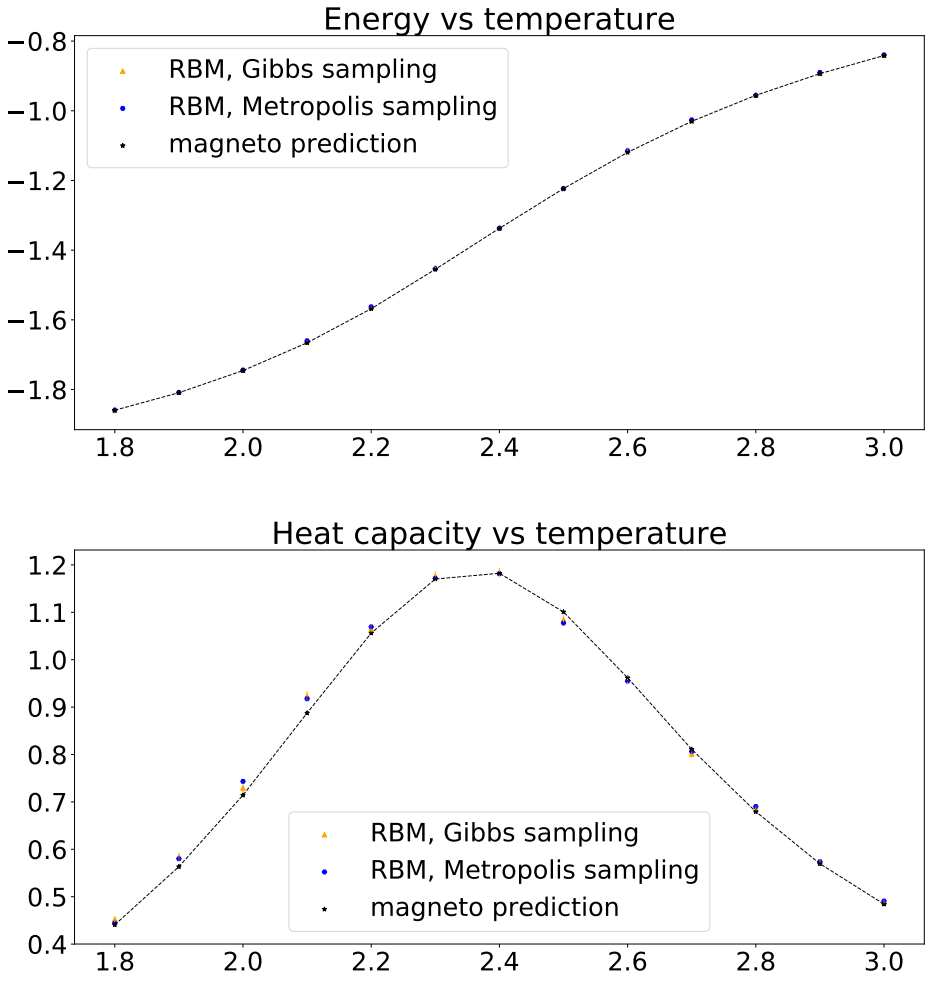

Figure 2.6 *Energy and heat capacity vs temperature. Observables are estimated on samples from performing Gibbs and Metropolis sampling on the trained RBMs. The model values are compared to the expected values from Magneto.*

e.g. see susceptibility at $T = 1.8$.

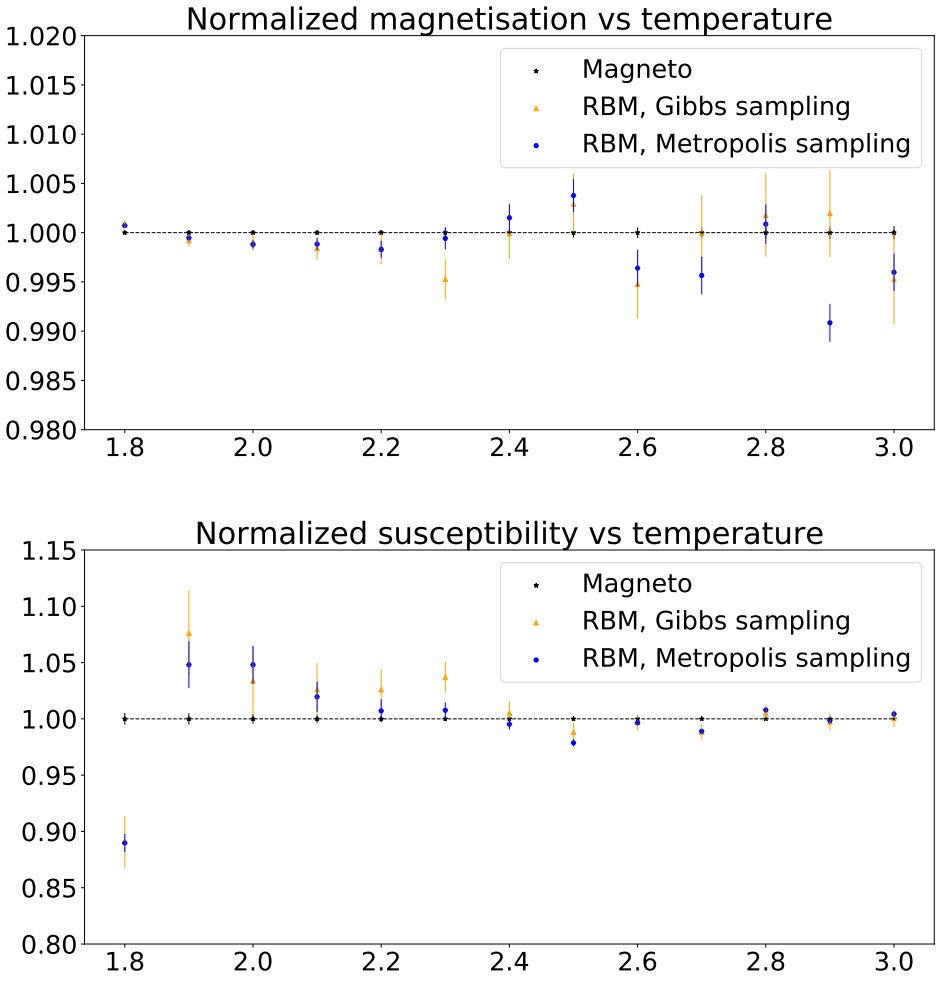

Figure 2.7 *Magnetisation and susceptibility, normalized by the expected magneto value, vs temperature.*

We conclude that the RBM is in general able to obtain precise values for the first moments of the underlying distribution at all temperature studied, however, for the second moments the agreement is not always as precise.

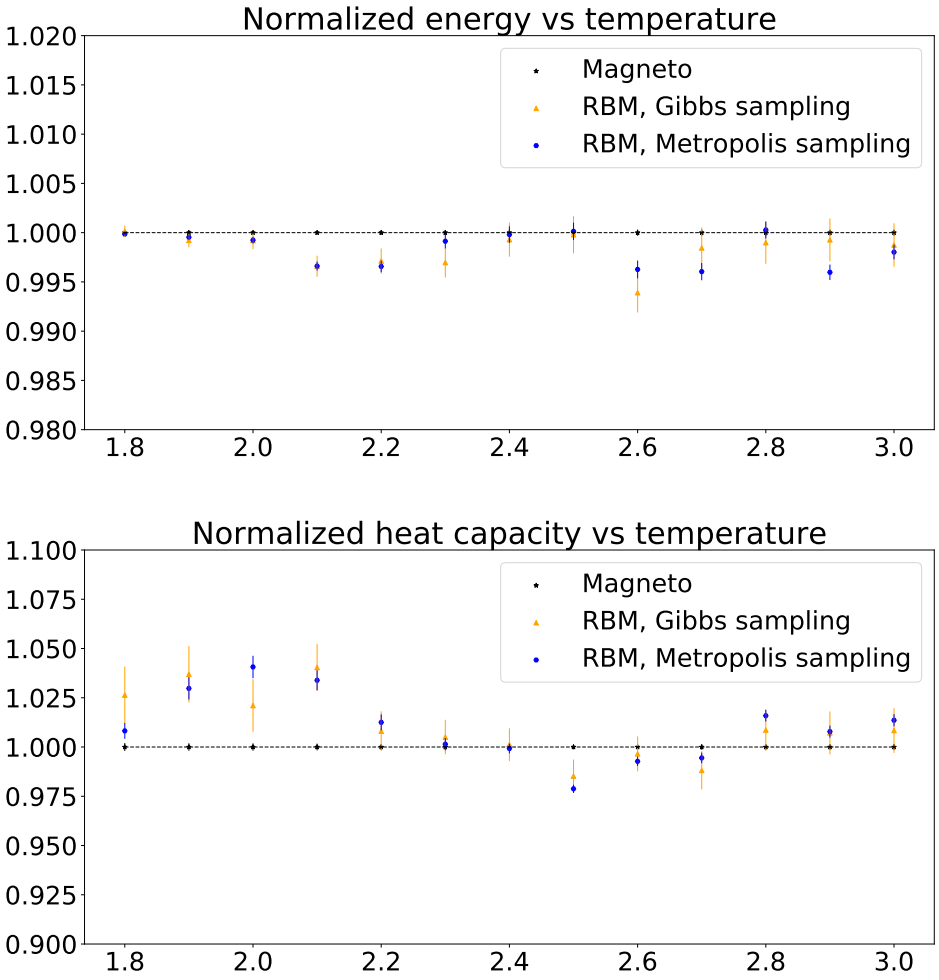

Figure 2.8 *Energy and heat capacity, normalized by the expected magneto value, vs temperature.*

2.5 Extracting couplings from the RBM

In the previous sections we described how to train an RBM for a 2D Ising model and assess the quality of the training by comparing the derived first and second moments, from the RBM, to the corresponding observables measured using the target distribution. This is a strong check that the machine has trained correctly, as first and second moments can always be measured from the training data directly. So far though, we haven't gained much from training the model and have only reproduced known results.

The predictive power of the machine, however, is in capturing the interactions between visible units, at every temperature. In this section we address how to extract the couplings between the visible units from the trained RBM, in the

case of a binary system *e.g.* the Ising model. Starting from an observation made in Chapter 16 of Ref. [53], we derive a closed form expression for the two-point coupling between the visible units, as a function of the parameters of the trained RBM, and compare the results with the two-points interactions expected from the Ising model. We then generalise the computation to higher order interactions, presenting a closed form expression for the tensor describing the n-point coupling between visible units learned by the machine, as a function of the learned RBM parameters.

In order to extract the coupling from the weight matrix of the trained RBM, w_{ij}, we expand the generating function in powers of v_j. Recall the free energy of the visible units, defined in Eq. 2.20

$$\begin{aligned} F(\mathbf{v}) &= \log \sum_{\mathbf{h}} e^{E(\mathbf{v},\mathbf{h}^{\ell}_{i,x})} \\ &= \sum_{i} \log \sum_{h_i} e^{-\sum_j b_j v_j - \sum_i c_i h_i - \sum_{i,j} h_i w_{ij} v_j} . \end{aligned} \tag{2.44}$$

Since the first term in $E(\mathbf{v}, \mathbf{h}^{\ell}_{i,x})$ is independent of the hidden layer, it can be brought outside the sum. This is the *visible bias* term. $F(\mathbf{v})$ then becomes,

$$\begin{aligned} F(\mathbf{v}) &= -\sum_j b_j v_j - \sum_i \log \sum_{h_i} e^{c_i h_i} e^{\sum_j h_i w_{ij} v_j} \\ &= -\sum_j b_j v_j - \sum_i \log \sum_{h_i} q(h_i) e^{t h_i} , \end{aligned} \tag{2.45}$$

where we have defined $t \equiv \sum_j w_{ij} v_j$ and $q(h_i) \equiv e^{c_i h_i}$. This allows us to introduce the cumulant generating function, similarly to Ref. [53]:

$$K_i(t) \equiv \log \sum_{h_i} q(h_i) e^{t h_i} = \sum_n \frac{\kappa_i^{(n)} t^n}{n!} , \tag{2.46}$$

where the nth cumulant $\kappa_i^{(n)} = \partial_t^n K_i(t)|_{t=0}$. Expanding the generating function

as a power series in n gives,

$$\begin{aligned} F(\mathbf{v}) &= -\sum_j b_j v_j - \sum_i \kappa_i^{(0)} - \sum_i \kappa_i^{(1)} t - \sum_i \frac{\kappa_i^{(2)} t^2}{2!} - \dots \\ &= -\sum_i \kappa_i^{(0)} - \sum_j \left(b_j + \sum_i \kappa_i^{(1)} w_{ij} \right) v_j + \\ &\quad - \frac{1}{2!} \sum_{j_1, j_2} \left(\sum_i \kappa_i^{(2)} w_{ij_1} w_{ij_2} \right) v_{j_1} v_{j_2} - \dots \ , \end{aligned} \tag{2.47}$$

where in the final line we have used the definition of t to rewrite $F(\mathbf{v})$ in terms of n-point interactions between the visible units. When the RBM has learned the physics of the Ising model, we would expect that $F(\mathbf{v}) = H_{\text{ising}}(\mathbf{v})$ up to a constant, as was discussed in Sec. 2.3. Since the two energies can differ by an overall constant without affecting the physical observables, we are not interested in $-\kappa_i^{(0)}$. However, the two energies are equal across states and we would expect that $F(\mathbf{v})$ and $H_{\text{ising}}(\mathbf{v})$ to be equal, order-by-order in $\mathbf{v}$. For the standard Ising interactions, with no external field,

$$H_{\text{Ising}}(\tilde{\mathbf{v}}) = \sum_{j_1, j_2} H_{j_1 j_2} \tilde{v}_{j_1} \tilde{v}_{j_2} \ , \tag{2.48}$$

where $\tilde{\mathbf{v}}$ describes the spin states, with each spin taking the value $+1$ or -1. $H_{j_1 j_2}$ is the appropriate Ising matrix with nearest neighbour interactions. In the Ising model used for this study, the two point interaction term is $-1/T$ where T is the temperature of the model. Therefore, $H_{j_1 j_2}$ is zero except for components which correspond to nearest neighbour interactions, which are equal to $\frac{1}{2T}$. This is shown pictorially in Fig. 2.9 for the case of an 8×8 lattice. The x and y axes represent spins on this two-dimensional lattice, ordered lexicographically. Notice the structure of couplings due to periodic boundary conditions.

States in the $\{-1, 1\}$ basis can be related to those in the $\{0, 1\}$ basis by the simple transformation $\tilde{\mathbf{v}} = 2\mathbf{v} - 1$. This allows us to rewrite the Ising Hamiltonian in the $\{0, 1\}$ basis as:

$$H_{\text{ising}}(\mathbf{v}) = 4 \sum_{j_1, j_2} H_{j_1 j_2} v_{j_1} v_{j_2} - 4 \sum_{j_1} \left(\sum_{j_2} H_{j_1 j_2} \right) v_{j_1} + \left(\sum_{j_1, j_2} H_{j_1 j_2} \right), \tag{2.49}$$

where the symmetric form of $H_{j_1 j_2}$ has allowed us to simplify the expression. The 2-point interaction in Eq. 2.49 and Eq. 2.47 can now be compared to extract the

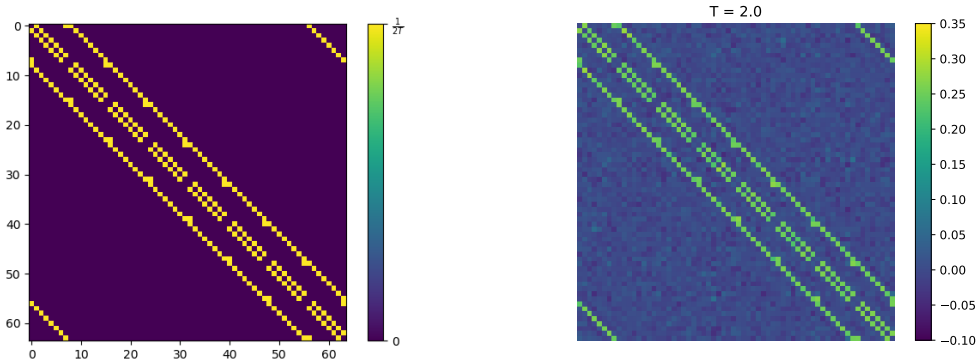

Figure 2.9 *The matrix of interactions, $H_{j_1 j_2}$, for an 8×8 Ising lattice with periodic boundary conditions as used in the generation of the training set (left) and the one learnt by the RBM at the end of the training (right). In this example T=2.0. The spins are labelled from 0 to 64 and the nearest neighbour structure is evident.*

coupling, $\frac{1}{2T}$, from the trained RBM.

There is however a subtlety which arises in the expansion of the cumulant generating function. We can see that in $\{0, 1\}$ basis:

$$v_j^n = v_j \quad , \quad n \in \mathbb{Z}^+ \cdot \tag{2.50}$$

This implies that higher order terms in n also contribute to the N_v-point function, where N_v is the number of *distinct* vertices. To make this statement more clear, let us write Eq. 2.47 in a more general form,

$$\begin{aligned} F(\mathbf{v}) = -\kappa_i^{(0)} - \sum_j \left(b_j + \sum_i \kappa_i^{(1)} w_{ij} \right) v_j + \\ - \sum_{n>1} \frac{1}{n!} \sum_{j_1 \ldots j_n} \left(\sum_i \kappa_i^{(n)} w_{ij_1} \ldots w_{ij_n} \right) v_{j_1} \ldots v_{j_n} \cdot \end{aligned} \tag{2.51}$$

In other words, n can be thought of as counting the powers of w_{ij}. For each n, the contributions from the sum over the visible units, $\sum_{j_1 \ldots j_n}$, can be grouped by the number N_v of distinct units being multiplied, which allows us to extract the N_v-point interactions. For example, using Eq. 2.50, we see that the numerical contributions to the 2-point interactions are terms proportional to $v_{j_1}^k v_{j_2}^{n-k} = v_{j_1} v_{j_2}$ where $j_1 \neq j_2$. The number of combinations of powers of v_{j_1} and v_{j_2} for a given n are simply the binomial coefficients. Therefore, 2-point contributions from all

orders in n can be written as:

$$\sum_{n>1} \frac{1}{2(n!)} \sum_{0<k<n} \sum_{j_1 \neq j_2} \left(\sum_i \kappa_i^{(n)} \binom{n}{k} w_{ij_1}^k w_{ij_2}^{n-k} \right) v_{j_1} v_{j_2} \ , \tag{2.52}$$

where the factor of two has been included to account for the double counting arising from the symmetry under the exchange of j_1 and j_2. Performing the sum over k gives

$$\frac{1}{2} \sum_{n>1} \frac{1}{n!} \sum_{j_1 \neq j_2} \left(\sum_i \kappa_i^{(n)} \left[(w_{ij_1} + w_{ij_2})^n - (w_{ij_1})^n - (w_{ij_2})^n \right] \right) v_{j_1} v_{j_2} \ . \tag{2.53}$$

Comparing this result to the Ising 2-point interaction in Eq. 2.49, provided that the RBM is properly trained, the Ising pair-wise coupling can be written as:

$$H_{j_1 j_2} = \frac{1}{8} \sum_{n>1} \frac{1}{n!} \sum_i \kappa_i^{(n)} \left[(w_{ij_1} + w_{ij_2})^n - (w_{ij_1})^n - (w_{ij_2})^n \right] \ . \tag{2.54}$$

Since $\kappa_i^{(n)} = \partial_t^n K_i(t)|_{t=0}$,

$$\begin{aligned} H_{j_1 j_2} =& \frac{1}{8} \sum_n \left(\frac{1}{n!} \sum_i \left[(w_{ij_1} + w_{ij_2})^n - (w_{ij_1})^n - (w_{ij_2})^n \right] \partial_t^n K_i(t)|_{t=0} \right) + \frac{1}{8} \sum_i K_i(0) \\ =& \frac{1}{8} \sum_i \left(e^{(w_{ij_1} + w_{ij_2})\partial_t} - e^{w_{ij_1}\partial_t} - e^{w_{ij_2}\partial_t} + 1 \right) K_i(t)|_{t=0} \ , \end{aligned} \tag{2.55}$$

where we evaluated the sum over n. Recognising the shift operator, $e^{a\partial_x} f(x) = f(a+x)$, we simplify the expression for the Ising pair-wise interaction

$$H_{j_1 j_2} = \frac{1}{8} \sum_i \Big(K_i(w_{ij_1} + w_{ij_2}) - K_i(w_{ij_1}) - K_i(w_{ij_2}) + K_i(0) \Big), \tag{2.56}$$

the closed form expression for $H_{j_1 j_2}$, including all order contributions in n. Substituting $K_i(t) = \log\left(1 + e^{c_i + t}\right)$, from Eq.(2.46)

$$H_{j_1 j_2} = \frac{1}{8} \sum_i \log \frac{(1 + e^{c_i + w_{ij_1} + w_{ij_2}})(1 + e^{c_i})}{(1 + e^{c_i + w_{ij_1}})(1 + e^{c_i + w_{ij_2}})} \ . \tag{2.57}$$

Using the trained RBMs, the coupling given by Eq. 2.56 was evaluated for temperatures ranging from $T = 1.8$ to $T = 3.0$. The results, presented in Fig. 2.11, have the same nearest neighbour structure as the Ising model used to train the

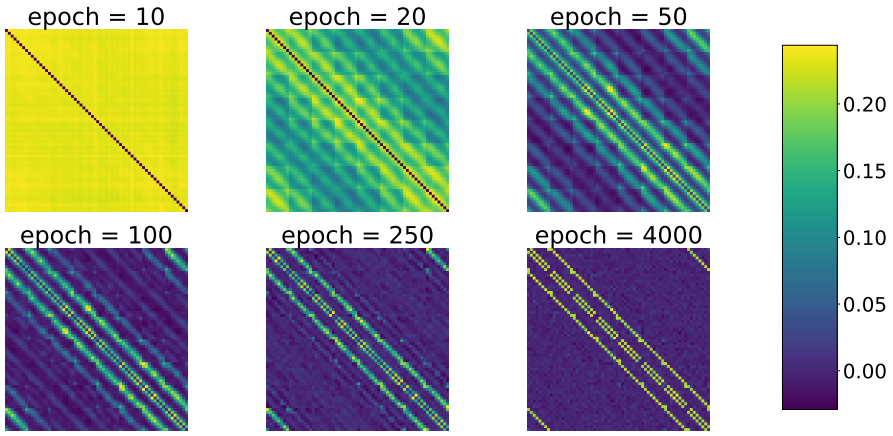

Figure 2.10 *$H_{j_1 j_2}$ extracted from RBMs at different stages of the training (10, 20, 50, 100, 250 and 4000 epochs), for $L^2 = 8 \times 8$, $h^2 = 8 \times 8$ and temperature $T = 2.2$. As the machine approaches the end of the training, the expected structure of Fig.2.9 becomes more and more evident.*

machines, in Fig. 2.9 with generic temperature T. In Fig. 2.12, we have plotted the corresponding histograms of the values $H_{j_1 j_2}$ at various temperatures. The expected bimodal structure of coupling matrices is observed, with most of the entries, *i.e.* those associated with the non-nearest neighbour spins interactions, being centred around zero, while the nearest neighbour interactions introduce a distinct second peak, around the desired value of the coupling. By taking the average and the standard deviation across the nearest neighbour diagonals we can compare the couplings extracted from the RBM against the exact values from the Ising model, as shown in Fig. 2.13. We see that the RBM predictions agrees with the analytical results within statistics.

It is also interesting to look at the change in the couplings along the training: this is shown in Fig. 2.10, where the 2-points interaction matrix for $T = 2.2$ is plotted at different stages of the training. The training algorithm starts by initialising the RBM visible-hidden interaction terms W_{ij} from a normal distribution centred around zero. The bias terms are also initialised to zero. In the early epochs, we observe no particular structure in the H_{j_1,j_2} interaction matrix. As learning progresses, more of the nearest neighbour structure appears, and spurious interactions disappear. Towards the final phases of the training, only the nearest neighbour structure remains, with the rest of the interactions being almost zero.

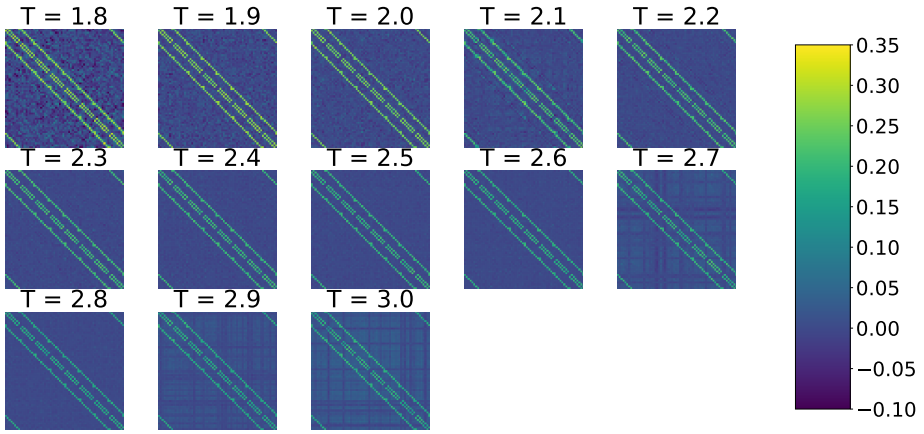

Figure 2.11 *The interaction matrix $H_{j_1 j_2}$ extracted from RBMs, with $L^2 = 8 \times 8$ and $h^2 = 8 \times 8$, trained at a temperature indicated above each subplot. Again, the spins are labelled from 0 to 64 and show the same structure as for the generic Ising training set in Fig. 2.9.*

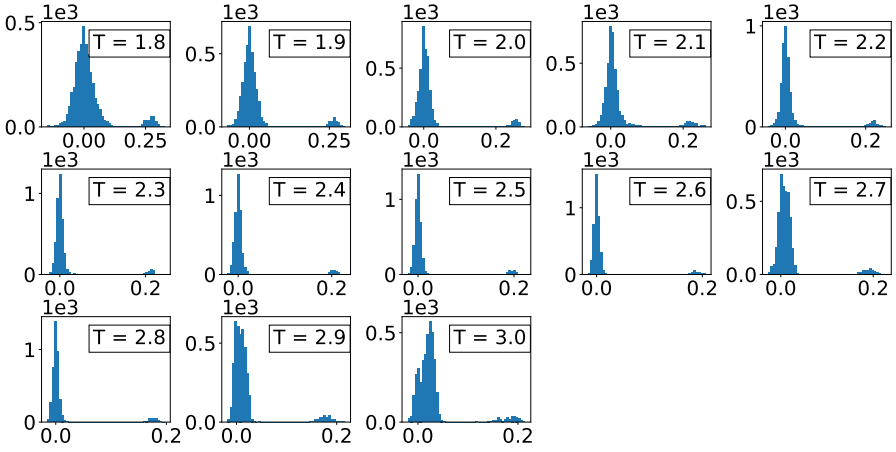

Figure 2.12 *Histograms of the entries of $H_{j_1 j_2}$ extracted from RBMs trained at a temperature indicated above each subplot. As it can be observed, there are always two peaks: The smaller peak, represents the number of nearest neighbour on the y-axis, with the value of the coupling indicated on the x-axis; the larger peak represent all other sites that are not nearest neighbours and are not expected to couple to each other, hence it being centred around zero on the x-axis.*

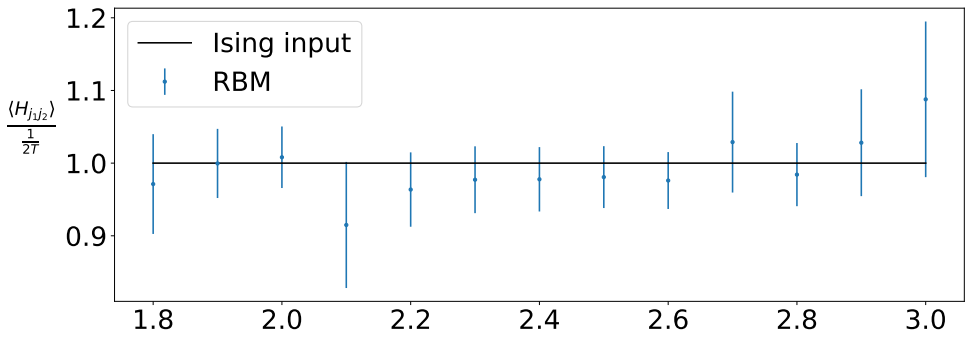

Figure 2.13 *The predicted 2-point interaction coupling, for $L^2 = 8\times8$ at different values of temperature, normalized by the corresponding true value $1/2T$. The coupling is extracted from the nearest neighbour diagonals observed in the interaction matrices, with its error bar computed by taking the standard deviation of the diagonal components. The predicted values agree with the expected ones within statistics.*

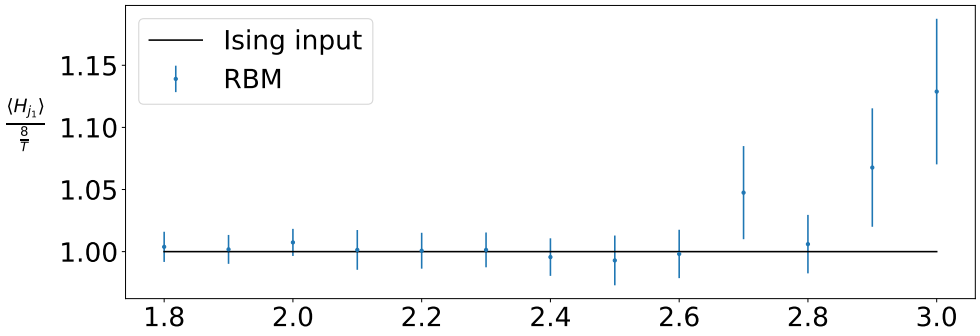

Figure 2.14 *The linear terms extracted by the trained machine, normalized by the corresponding true value $8/T$. All predicted values are compatible with the true ones within 2σ. The largest discrepancies are observed for the highest temperatures, where there is an indication that the training hasn't fully converged. As discussed in Table 2.3, the training parameters were the same for all models with $T \geq 2.2$ and here we have an indication that the parameters should have been further tuned for the highest temperature models.*

Let us now return to Eq. 2.51 and collect terms linear in v_j. The terms contributing are those with $j_1 = j_2 = \ldots = j_n$, giving,

$$\sum_j J_j v_j = -\sum_j \left(b_j + \sum_i \kappa_i^{(1)} w_{ij} \right) v_j - \sum_{n>1} \frac{1}{n!} \sum_j \left(\sum_i \kappa_i^{(n)} \left(w_{ij} \right)^n \right) v_j. \quad (2.58)$$

The vector J_j can be read from the linear term in Eq. 2.49,

$$J_j = 4 \sum_i H_{ij} = \frac{8}{T} \ , \quad (2.59)$$

where we have used $\sum_i H_{ij} = 2/T$, for every component i in the Ising 2-point interaction matrix H_{ij}. The second term on the right hand side of the Eq. 2.58 can be treated in the same way as the 2-point interaction described above above,

$$\sum_{n>1} \frac{1}{n!} \sum_j \left(\sum_i \kappa_i^{(n)} \left(w_{ij} \right)^n \right) v_j = \sum_j \sum_i \left[K_i \left(w_{ij} \right) - K_i \left(0 \right) \right] v_j - \sum_j \sum_i \kappa_i^{(1)} w_{ij} v_j \ . \quad (2.60)$$

It then follows that,

$$J_j = -\left(b_j + \sum_i K_i \left(w_{ij} \right) - \sum_i K_i \left(0 \right) \right) = -\left(b_j + \sum_i \log \left(\frac{1 + e^{c_i + w_{ij}}}{1 + e^{c_i}} \right) \right). \quad (2.61)$$

This allows us to extract the value of an external magnetic field, if present, from the parameters of the trained RBM. The original training set was generated without any external magnetic fields, therefore, if working in the $\{-1, 1\}$ basis, we do not expect to observe any linear term from the trained machine. However, since we trained the RBM in the $\{0, 1\}$ bases, we expect that $J_j = 8/T$ The linear term, J_j extracted from the trained machine are plotted in Fig. 2.14, where the error bar is obtained by computing the standard deviation of the entries J_j. Again, we see how these values are compatible with the expected ones within 1σ for most temperatures, and 2σ for higher temperatures. The linear term appears to be systematically overestimated for higher temperatures, which is probably due to the training parameters of the highest temperature models needing finer tuning. If this is some kind of under-learning, then the systematic shift might be due to the initialisation of the model parameters.

The 3- and 4-point interactions can also be calculated and plotted. This is mainly used as a sanity check to confirm that the RBM has not incorrectly learned unexpected additional interactions. For the 3 point-interactions, we want to

include all combinations which leave three distinct indices, similar to Eq. 2.52 we get,

$$\sum_{n>2} \frac{1}{6(n!)} \sum_{0<l<(n-k)} \sum_{0<k<(n-1)} \sum_{j_1 \neq j_2 \neq j_3} \Big(\sum_i \kappa_i^{(n)} \binom{n}{k} \binom{n-k}{l} w_{ij_1}^k w_{ij_2}^l w_{ij_3}^{n-k-l} \Big) v_{j_1} v_{j_2} v_{j_3} \ . \tag{2.62}$$

Summing over k and l gives

$$\begin{aligned} \sum_{n>2}^{\infty} \frac{1}{6(n!)} \sum_{j_1 \neq j_2 \neq j_3} \Big(\sum_i \kappa_i^{(n)} \big[& (w_{ij_1} + w_{ij_2} + w_{ij_3})^n + \\ & - (w_{ij_1} + w_{ij_2})^n - (w_{ij_1} + w_{ij_3})^n - (w_{ij_2} + w_{ij_3})^n \\ & + (w_{ij_1})^n + (w_{ij_2})^n + (w_{ij_3})^n \big] \Big) v_{j_1} v_{j_2} v_{j_3} \ . \end{aligned} \tag{2.63}$$

If we account for terms corresponding to $n = 0, 1, 2$ then the sum over n can be evaluated as before, giving

$$\begin{aligned} \frac{1}{6} \sum_{j_1 \neq j_2 \neq j_3} \sum_i \Big(& K_i(w_{ij_1} + w_{ij_2} + w_{ij_3}) + \\ & - K_i(w_{ij_1} + w_{ij_2}) - K_i(w_{ij_1} + w_{ij_3}) - K_i(w_{ij_2} + w_{ij_3}) + \\ & + K_i(w_{ij_1}) + K_i(w_{ij_2}) + K_i(w_{ij_3}) + K_i(0) \Big) v_{j_1} v_{j_2} v_{j_3} \ . \end{aligned} \tag{2.64}$$

We can then write the interaction tensor explicitly as

$$\frac{1}{6} \sum_i \log \frac{(1 + e^{c_i + w_{ij_1} + w_{ij_2} + w_{ij_3}})(1 + e^{c_i + w_{ij_1}})(1 + e^{c_i + w_{ij_2}})(1 + e^{c_i + w_{ij_3}})(1 + e^{c_i})}{(1 + e^{c_i + w_{ij_1} + w_{ij_2} +})(1 + e^{c_i + w_{ij_1} + w_{ij_3}})(1 + e^{c_i + w_{ij_2} + w_{ij_3}})} \ . \tag{2.65}$$

One can observed the pattern emerging from (2.57) and (2.65) and write down a general form for the N-point interaction tensor

$$\frac{1}{N!} \sum_{l=0}^{N} (-1)^l \sum_{\alpha_1 < \ldots < \alpha_{N-l}} K_i(w_{i,j_{\alpha_1}} + \ldots + w_{i,j_{\alpha_{N-l}}}) \tag{2.66}$$

where $\{\alpha_i\} = [1..N]$ and the inner sum represents the $\binom{N}{N-l}$ combinations of choosing $N - l$ unique indices from $\{\alpha_i\}$. Using Eq. 2.66, histograms for the tensors for the elements of the 3- and 4-point interaction can then be plotted for the trained RBMs, to check if any spurious higher order interactions were learned.

It is expected that the histograms have a single peak at zero. These histograms are presented in Fig. A.13 and Fig. A.14 of App. A.5, respectively.

2.6 Conclusions

We have trained several RBMs on 1- and 2-dimensional Ising models at various values of temperature. The training procedure in each case has been discussed in detail. We have used five different criteria to test whether the learning process has been successful. The first four are measurements of the loss function, reconstruction error, free energy and log-likelihood throughout the training procedure. The measurement of the latter, which is the quantity that is being maximised in the algorithm, involves the estimation of the partition function of the model, which is computed using annealed importance sampling. The fifth and last criterion, is the measurements of the first and second moments of the distribution, as given by the RBM, and comparing them to those obtained directly from the training set. These five criteria are essential to the RBM being trained correctly. Hence, we have provided a generic prescription in training an RBM on a binary model, from our experience.

Moreover, we have used the RBM to predict the interaction couplings between the Ising spins by re-summing the cumulant generating function. The re-summations can be performed for any model with binary $\{0, 1\}$ states. The ability to extract the couplings exactly can be useful in studying the relation between the RBM and the renormalization group (RG), as first noted by Ref. [49]. In this paper, we have also demonstrated the difficulty in training an RBM with smaller number of hidden nodes as compared to the visible. The numerical verification of RG is therefore left for future work. Another application of predicting the couplings is for the case where the interactions between the visible nodes of the RBM are unknown. In general, interactions between the visible units are more complicated than simple pair-wise interactions and as such cannot be extracted directly from the data with conventional methods. The RBM has no such assumption, as there are indirect all-to-all connections between the visible nodes, via the hidden layer. As the machine learns from the data, some of these connections turn-off while others increase in strength. This makes the RBMs a powerful tool for predicting complex connections between the visible nodes and their relative strength.

Chapter 3

Normalizing flows

3.1 Introduction

Following on from the previous section, where we saw that a generative model could learn an approximation of the distribution which generates a set of training examples. The interesting feature of the trained RBM was it could be used to make a prediction for the couplings between visible units, which would be particularly useful in systems where the underlying Hamiltonian was unknown.

In this work we change tack and instead wonder if the generative model can be used as an efficient sampler, either replacing or augmenting a traditional MCMC method. It's clear that in order to realise this, we will require some different techniques because, as we saw in Chapter 2, the RBM required a large amount of pre-existing training examples which have to be sampled before the model can be trained. This kind of approach doesn't appear to offer any upside, since if we can already produce a sample large enough to train the model then it's unclear what the benefit of training the model will be. In sec. 3.3.1 we will show how a certain class of models, *normalizing flows*, can bypass this issue in the case that the target distribution has a known Boltzmann weight.

We can generalise the estimation of observables for the Ising model to other lattice field theories by rephrasing Eq. 2.36 as

$$\langle \mathcal{O} \rangle = \frac{1}{\mathcal{Z}} \int \mathcal{D}\phi \, e^{-S(\phi)} \mathcal{O}(\phi) \,, \tag{3.1}$$

where

$$\mathcal{D}\phi = \prod_{x\in\Lambda} d\phi_x \,, \tag{3.2}$$

the Euclidean path integrals have been discretised onto a space-time lattice Λ. $\mathcal{O}(\phi)$ is a generic observable defined for the field configuration ϕ, and the (Euclidean) action $S(\phi)$ encodes all of the dynamics and interactions of the fields.

In practical terms, these integrals are evaluated as expectation values over a sample, as with the Ising model,

$$\bar{\mathcal{O}} = \frac{1}{|\Phi|}\sum_{\phi\in\Phi} \mathcal{O}(\phi) \,, \tag{3.3}$$

where the representative sample Φ is comprised of field configurations drawn from a statistical ensemble with Boltzmann factor $e^{-S(\phi)}$. The error on this estimator scales as $1/\sqrt{N_{\text{eff}}}$, where the effective sample size N_{eff} reaches a maximum value of $|\Phi|$, the number of configurations in the sample, in the absence of correlations between configurations.

As mentioned in Sec. 2.4.3, sampling from high-dimensional distributions can be achieved with Markov Chain Monte Carlo (MCMC) methods, such as the Metropolis algorithm. However, the configurations in the resulting sequence are indeed correlated, and the effective sample size is diminished by a factor of twice the *integrated autocorrelation time*,

$$\tau_{\text{int},\mathcal{O}} = \frac{1}{2} + \sum_{t=1}^{\infty} A(t) \,, \tag{3.4}$$

where we have integrated over the autocorrelation time, which was defined for the magnetisation in Eq. 2.43. Here we generalise the definition of the autocorrelation time

$$A(t) = \frac{\Gamma_{\mathcal{O}}(t)}{\Gamma_{\mathcal{O}}(0)} \,, \tag{3.5}$$

defined for each observable in terms of its autocorrelation function $\Gamma_{\mathcal{O}}(t) = \langle \mathcal{O}(\phi^{(n+t)})\mathcal{O}(\phi^{(n)})\rangle - \langle\mathcal{O}\rangle^2$, where t represents a number of steps separating pairs of field configurations in a Markov chain, and n is arbitrary provided the process has equilibrated to its stationary distribution [63]. Practically speaking, the autocorrelation time is estimated over a finite sample, as in Eq. 2.43.

Under normal conditions this issue is manageable. Most MCMC algorithms, however, suffer from an acute condition known as *critical slowing down* associated

with a quite catastrophic reduction in their sampling efficiency as the system under study approaches a critical point [64]. Critical slowing down typically manifests as a power-law scaling of the integrated autocorrelation time with the system's correlation length ξ,

$$\tau_{\text{int},\mathcal{O}} \propto \xi^{z_\mathcal{O}} \,, \tag{3.6}$$

where ξ (in lattice units) diverges as we take the continuum limit of our lattice field theory.

Algorithms based on local updates [38, 39, 65] have lower limit of $z_\mathcal{O} = 1$ owing to the maximum speed of information propagation, but in the absence of very careful tuning [66] they typically exhibit $z_\mathcal{O} = 2$ scaling, corresponding to diffusive information transport. Furthermore, there is substantial evidence that the picture is even worse when considering theories which possess non-trivial topology in the continuum limit [67–71], including QCD itself [72, 73]. As the continuum limit is approached, the rapid increase in free energy barriers between topological sectors can result in $z_\mathcal{O} > 2$ for topological observables, and potentially even exponential scaling [74].

Collective update algorithms should, in principle, fare much better, because they are not restricted to local (e.g. random-walk or Hamiltonian) dynamics. Indeed, in Chapter 2 we already made use of the Swendsen-Wang (SW) algorithm [62] to produce the training data, which is based on collective updates. Whilst algorithms based on collective updates have been devised for certain systems [75–78], there is no general-purpose collective update algorithm, and critical slowing down remains an unsolved problem in the majority of cases, including lattice QCD.

This motivates augmenting the pre-exisitng MCMC techniques with tools from ML. Techniques which are either based on the models discussed in Chapter 2 or similar but are comprised of more hidden layers, *deep generative models* (DGMs) [79, 80], seem like comprising candidates for this task. As we have seen, once trained, the DGMs can be directly sampled from in order to generate configurations for physical models. Already, a number of prototypical hybrid algorithms have been proposed in which DGMs either guide or replace traditional MCMC update procedures [1, 81–95].

Normalizing flows [96–98] are a class of DGM which model the distribution of interest by learning an invertible map from a set of *latent* variables whose distribution is much easier to sample from. Typically, the map is built out of a sequence of relatively simple element-wise transformations [99, 100]. The capacity

to model complex, correlated probability distributions arises due to the fact that the parameters of these transformations are generated by neural networks that take the variables themselves as inputs. Put another way, the process of training such a model is an encoding of the correlations between the degrees of freedom in the path integral into the weights and biases of these neural networks.

Normalizing flows remind us of Lüscher's *trivializing map* [101], in which the theory is mapped onto one where the field variables decouple, through an invertible field transformation whose Jacobian negates the weighting effect of the action. Lüscher provided a power-series expression for the generators of a class of flows which trivialize lattice gauge theories, although only the first two terms in this series are tractable in practice, and additional finite-step errors are accumulated through numerical integration of the flow. Unfortunately, the degree to which the result of this procedure approximates a trivializing map proved insufficient to improve the scaling of integrated autocorrelation times [102]. However, there has recently been a renewed interest in the potential to construct approximately trivializing maps with additional leverage provided by modern machine learning techniques.

In Secs. 3.2 and 3.3 we outline a procedure in which a normalizing flow generates statistically independent field configurations that act as proposals for the Metropolis-Hastings algorithm. From a theoretical perspective, this approach has the potential to become more efficient than traditional sampling, since the statistical efficiency of the sampling algorithm is decoupled from the correlation length of the system. The caveat is that, somehow, the costs associated with the highly non-trivial task of sampling from the path integral of an interacting field theory are transferred to the training of the model. Therefore, to answer the question of whether a generative sampling algorithm can be expected to outperform traditional methods is a matter of understanding how these training costs scale as the continuum limit is approached.

We compare our results to a seminal paper which first demonstrated that the procedure just described is a viable approach to sampling in lattice field theory [1]. In their proof of principle study, which focused on two-dimensional scalar ϕ^4 theory on small lattices (up to 14^2 sites), the normalizing flow was a sequence of element-wise affine transformations parameterised by neural networks. Here, we continue the same thread, with the aim of establishing how well this approach scales to larger lattices (up to 20^2 sites). Finding that the recipe used in Ref. [1] yields fairly inefficient representations of trivializing maps for the particular theory of

interest, we make a number of adjustments; most importantly, we introduce a more expressive transformation based on a *spline*, and replace the deep neural networks by networks with a single hidden layer that is rather narrow. These changes facilitate a large reduction in the computational cost required to train models of equivalent quality. Finally, we quantify the scaling of our models towards the continuum limit using hardware-independent metrics — the Metropolis-Hastings acceptance rate, the number of trainable parameters in the models, and the total number of field configurations generated during the training phase.

3.2 Sampling in lattice field theory

The problem we are trying to solve can be phrased as follows: we would like to generate samples $\Phi = (\phi^{(1)}, \phi^{(2)}, \ldots, \phi^{(|\Phi|)})$ of the discrete random field $\{\phi_x \mid x \in \Lambda\} \equiv \phi \in \mathcal{M}^{|\Lambda|}$, where $|\Lambda|$ is the number of sites on the lattice and $\mathcal{M}^{|\Lambda|}$ is a direct product called the field space, that are representative of the lattice field theory we seek to study. By 'representative' we mean that the probability of a particular configuration appearing in the sample is to be proportional to its Boltzmann weight,

$$p(\phi) \equiv \frac{e^{-S(\phi)}}{\mathcal{Z}} \,. \tag{3.7}$$

We will refer to $p(\phi)$ as the *target density*.

3.2.1 Markov Chain Monte Carlo

MCMC sampling methods work by generating a sequence of independent transitions $\phi^{(n)} \to \phi^{(n+1)}$ which construct a Markov chain $(\phi^{(n)})_{n=1}^{N}$. Thus, implicit in any MCMC method is a transition kernel $W(\phi \to \phi')$, which is required to have a stationary distribution that is equal to the distribution from which we wish to sample, implying

$$\int \mathcal{D}\phi \, p(\phi) W(\phi \to \phi') = p(\phi') \,. \tag{3.8}$$

If $W(\phi \to \phi')$ is also ergodic, then the stationary distribution is unique and the Markov chain is guaranteed to converge to $p(\phi)$ [103]. However, this does not imply that any finite section of the chain is representative of $p(\phi)$, since the process of generating configurations via a transition kernel introduces autocorrelations. In practice, this results in statistical errors that scale as $(2\tau_{\text{int},\mathcal{O}} N)^{-1/2}$ rather

than $N^{-1/2}$, leading to a trade-off between algorithmic efficiency — the amount of effort taken to generate a transition — and statistical efficiency — how many transitions are required to produce a statistically independent configuration.

The approach of the Metropolis-Hastings algorithm [38, 39] is to generate configurations ϕ' via a distribution $q(\phi' \mid \phi)$ which is easy to sample from, and accept or reject these proposals based on an acceptance probability $A(\phi \to \phi')$ such that $W(\phi \to \phi') = q(\phi' \mid \phi)A(\phi \to \phi')$ and detailed balance is satisfied. The standard choice is the 'Metropolis test',

$$A(\phi \to \phi') = \min\left(1, \frac{q(\phi \mid \phi')}{q(\phi' \mid \phi)} \frac{p(\phi')}{p(\phi)}\right), \tag{3.9}$$

which is a more general version of the Metropolis test in Sec. 2.4.3, in which $q(\phi' \mid \phi)$ was uniform. Importantly, Eq. 3.9 does not require the calculation of normalizing factors. A rejection of the proposal corresponds to a duplication of the current state in the chain. Thus, the Markov chain can be seen as a reweighting of the set of proposals in which the configurations pick up integer weights. The proposal distribution $q(\phi' \mid \phi)$ can be anything which guarantees ergodicity of $W(\phi \to \phi')$, and it is sufficient for it to have non-zero density everywhere on $\mathcal{M}^{|\Lambda|}$ [104].

The Metropolis-Hastings (MH) is demonstrated by the following snippet of Python pseudo-code. `generator` yields proposals drawn from $q(\phi' \mid \phi)$, and `acceptance` is a function which evaluates (3.9).

Metropolis-Hastings Algorithm (Python)

```
chain = []
current = initial = next(generator) # initialise
for n in range(N):
    proposal = next(generator)
    prob = acceptance(current, proposal)
    r = rand() # random uniform number [0, 1]
    if r < prob:
        chain.append(proposal)
        current = proposal
    else:
        chain.append(current)
```

The MH algorithm is completely agnostic towards the process through which proposals are generated provided any 'selection bias' is properly accounted for by the factor $q(\phi \mid \phi')/q(\phi' \mid \phi)$. This makes it very appealing as a kernel around which to construct collective updates algorithms.

The complexity of this approach is introduced due to the rapidly increasing sparsity of $p(\phi)$ as the number of degrees of freedom is increased when we move towards the continuum, which puts extremely stringent constraints on how proposals may be generated if we are to sample the path integral in an acceptable amount of time. The two main approaches to this problem are

- *Local updates*: Generate proposals that are close to the current configuration by updating individual lattice sites. Changes in $p(\phi)$ can be made arbitrarily small by tuning the step size so as to yield a desired acceptance rate.
- *Hybrid Monte Carlo*: Generate proposals by numerically integrating a fictitious Hamiltonian system, and by doing so update all of the lattice sites. In this case it is the number of integration steps that must be balanced against the acceptance rate.

Both of these methods become less efficient when we take the continuum limit, as we are forced to trade down on step size to keep the acceptance rate reasonably high, meaning each statistically independent configuration requires more effort to produce. This is what is meant by critical slowing down, and the decline in statistical efficiency is quantified by the dynamical critical exponent in Eq. (3.6).

3.2.2 A generative approach to global updates

Given what we have already seen about generative models in Chapter 2, it seems feasible that a generative model could take on the role of the generator in the MH algorithm.

For reasons which will shortly become clear, we will focus the following discussion on parametric DGMs with an explicit probability density, $\tilde{p}_\theta(\phi)$, where the subscript θ collectively labels the model's parameters. The intention is to construct a model and identify a set of parameters such that the approximation $\tilde{p}_\theta(\phi) \approx p(\phi)$ is a 'good' one.

The notion of a 'good approximation' is made quantitative through the Kullbach-Leibler divergence [105],

$$D_{\mathrm{KL}}(p \,||\, \tilde{p}_\theta) = \int \mathcal{D}\phi \, p(\phi) \log \frac{p(\phi)}{\tilde{p}_\theta(\phi)} \,. \tag{3.10}$$

In the present work, the loss function will be (a variant of) the Kullbach-Leibler divergence, and the goal of training will be to find the set of parameters θ^* which satisfy

$$\theta^* = \arg\min_\theta D_{\mathrm{KL}}(p \,||\, \tilde{p}_\theta). \tag{3.11}$$

As discussed in Sec. 3.2.1, problems will arise if we are unable to fit the target distribution well enough, i.e. $\tilde{p}_\theta(\phi) = p(\phi)$, which would imply that the samples generated by the DGM are not truly representative of the field theory, with discrepancies between $\tilde{p}_\theta(\phi)$ and $p(\phi)$ manifesting as biases in expectation values. Yet it is possible to exactly correct for these biases through reweighting or a Metropolis step, provided we have access to $\tilde{p}_\theta(\phi)$. Hence, as well as restricting ourselves to models with an *explicit* density function, we will also demand that $\tilde{p}_\theta(\phi)$ is *tractable*, by which we mean it is given exactly by a closed-from expression computable in polynomial time (to be scalable) and whose repeated evaluation (for us, $10^5 - -10^9$ times during training) does not constitute an unacceptably large overhead.

Consider a variant of the MH algorithm in which `generator` is a DGM equipped with an explicit and tractable density that is capable of generating independent configurations with probability

$$q(\phi' \mid \phi) = \tilde{p}_\theta(\phi') \,. \tag{3.12}$$

If the model were a perfect approximation such that $\tilde{p}_\theta(\phi) = p(\phi)$ for all ϕ, then $A(\phi \to \phi') = 1$ identically and 100% of proposals would be accepted. In a more realistic situation where there are discrepancies, the inefficiency of generating proposals with a probability proportional to $\tilde{p}_\theta(\phi)$ rather than $p(\phi)$ manifests itself through the multiplicities in the Markov chain due to rejections, which are in turn measurable as autocorrelations. However, if proposals are drawn independently, then rejections are the *only* source of autocorrelation. As explained in Reference

[1], the autocorrelation at separation t is given, *for all observables*, by

$$\begin{aligned}\frac{\Gamma(t)}{\Gamma(0)} &= \Pr(t \text{ consecutive rejections}) \\ &= \mathbb{E}_{\phi\sim\tilde{p}_\theta}\left[\left(\mathbb{E}_{\phi'\sim\tilde{p}_\theta}\left[1 - A(\phi \to \phi')\right]\right)^t\right] . \end{aligned} \tag{3.13}$$

This is an extremely appealing feature that is not present in traditional algorithms, where random-walk dynamics combined with free energy barriers can lead to the decoupling of autocorrelation times for different observables [69].

Since Equation (3.13) is strictly larger than the average rejection rate $1 - \mathbb{E}_{\phi,\phi'\sim\tilde{p}_\theta}\left[A(\phi \to \phi')\right]$ raised to the t-th power, a lower bound on the integrated autocorrelation time can be given in closed form by a geometric series,

$$\tau_{\text{int}} \geq \frac{1}{\mathbb{E}_{\phi,\phi'\sim\tilde{p}_\theta}\left[A(\phi \to \phi')\right]} - \frac{1}{2} . \tag{3.14}$$

This expression is not particularly useful per se, but we will be interested in how close to this lower bound the actual integrated autocorrelation falls.

Although this is not the approach we will take, reweighting can instead be done at the level of computing ensemble averages [90], through a change of measure in (3.1) to $\mathcal{D}\phi\,\tilde{p}_\theta(\phi)w(\phi)$, where the reweighting factor $w(\phi) \equiv p(\phi)/\tilde{p}_\theta(\phi)$ is the same factor used in the MH acceptance test. The mean estimator (3.3) then reads

$$\bar{\mathcal{O}} = \frac{\sum_{\phi\in\Phi} w(\phi)\mathcal{O}(\phi)}{\sum_{\phi\in\Phi} w(\phi)} . \tag{3.15}$$

Of course, while this approach makes use of all of the generated configurations, there is still a price to be paid for drawing samples from $\tilde{p}_\theta(\phi)$ rather than $p(\phi)$; the weights ensure that the number of configurations yielding non-negligible contributions to the sum drops rapidly as the approximation $\tilde{p}_\theta(\phi) \approx p(\phi)$ degrades. As remarked on in Reference [91], this a posteriori reweighting approach is appealing if $\mathcal{O}(\phi)$ is cheap to compute relative to the cost of generating configurations from the model.

3.3 Normalizing flows

As the bases of this work, we define a normalizing flow as a bijective mapping

$$\begin{aligned} f_\theta : \mathcal{M}^{|\Lambda|} &\to \mathcal{M}^{|\Lambda|} \\ z &\mapsto \phi = f_\theta(z) \end{aligned}$$

between *latent* random variables, $z \sim r(z)$, and field configurations, $\phi = f_\theta(z) \sim \tilde{p}_\theta(\phi)$. We should note, however, that field configurations generated by the flow model are proposals for the MH algorithm, and will not be used directly in the estimation of observables. Normalizing flows are traditionally constructed by combining multiple, relatively simple transformations g_i through function composition: $f_\theta \equiv g_I \circ g_{I-1} \circ \ldots \circ g_2 \circ g_1$.

We will immediately restrict ourselves to the special case of $\mathcal{M} = \mathbb{R}$, which applies to scalar ϕ^4 theory. There has been discussion of flows on non-Euclidean manifolds [106, 107], but this is beyond the scope of this work. The density associated with the candidate field configurations is given by a change of variables, involving the Jacobian determinant,

$$\tilde{p}_\theta\big(f_\theta(z)\big) = r(z)\left|\det \frac{\partial f_\theta(z)}{\partial z}\right|^{-1}. \tag{3.16}$$

We will draw latent variables from an uncorrelated Gaussian distribution,

$$r(z) = \prod_{x\in\Lambda} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-z_x^2/(2\sigma^2)}\,, \tag{3.17}$$

which one may interpret, in the spirit of Refs. [101, 102], as a trivial limit of ϕ^4 theory.

In principle one could put more effort into generating latent variables that reduce the workload for the flow, provided that the distribution of the latent variables is known up to an overall normalization. However, with building a fast and efficient sampler in mind, we will avoid over-engineering the latent variable for now. Also it's key that samples from the latent distribution are i.i.d, in order to guarantee that the candidate field proposals are also independent, which is easy to achieve with a sample latent variable distribution.

Although the bijective construction is not the most flexible a priori, normalizing flows have several advantages over other DGMs. Crucially, it is straightforward in principle to ensure that the density is tractable, by choosing a map whose Jacobian determinant $\det \partial f_\theta(z)/\partial z$ is tractable. It is this feature that provides us with a means of guaranteeing convergence to the correct target density through the Metropolis test. Additional benefits relate to the training, which is discussed in the next subsection. Finally, as an added bonus, the intermediate states of the flow correspond to valid probability densities in their own right, from which we can draw samples. In this sense, there may be scope to interpret the distributions of the intermediate variables, work which would be along a similar line to the extraction of the couplings in Chapter 2 or relating DGMs to RG transformations [108–112]. We leave this as the focus of future work on normalizing flows.

3.3.1 Training a flow model

Since normalizing flows are differentiable by construction, they can be trained using standard gradient-based optimisation algorithms. The algorithm used in this work is a variant of SGD, which incorporates momentum, called ADAM [113]. The conventional approach to training is to expose the model to a set of data drawn from the distribution of interest, $p(\phi)$, via a separate process and tune the parameters of the model in order to optimise some loss function. If we were to take the conventional approach here, the set training configurations would be divided into minibatches, passed through the layers of the flow model in the reverse direction, and the resulting variables $f_\theta^{-1}(\phi)$ used to estimate the following loss function:

$$\hat{D}_{\mathrm{KL}}(p \,||\, \tilde{p}_\theta) = \mathbb{E}_{\phi\sim p(\phi)} \left[-\log r\big(f_\theta^{-1}(\phi)\big) - \log \left| \det \frac{\partial f_\theta^{-1}(\phi)}{\partial \phi} \right| \right] + \text{irrelevant terms}\,, \tag{3.18}$$

where the expectation value $\mathbb{E}_{\phi\sim p(\phi)}$ is computed by taking the average over the batch. Eq. 3.18 is an estimator for the Kullbach-Leibler (KL) divergence defined in Eq. 3.10, up to an unknown self-information term, $\mathbb{E}_{\phi\sim p(\phi)} \log p(\phi)$, that does not depend on the model's parameters and is therefore irrelevant for the purposes of optimisation. Note that minimising the KL divergence in Eq. 3.18 is equivalent to maximising the likelihood as was done with the RBM once the irrelevant terms which do not depend on the model are discarded.

For our purposes this strategy is clearly not satisfactory since the problem has gone

full circle; the ability to train models would then be tied to the ability to generate a large representative sample of configurations to act as training data, which is exactly what we are prevented from doing by critical slowing down. Thankfully, an alternative path presents itself in the typical scenario where we are interested in sampling from a theory for which $S(\phi)$ is completely specified. In this training paradigm, one takes the *reverse* of Eq. 3.10, noting that the KL divergence is not symmetric

$$D_{\mathrm{KL}}(\tilde{p}_\theta \,||\, p) = \int \mathcal{D}\phi\, \tilde{p}_\theta(\phi) \log \frac{\tilde{p}_\theta(\phi)}{p(\phi)}\,. \tag{3.19}$$

This allows us to define an loss function that can be minimised using estimates based on configurations generated exclusively by the model:

$$\hat{D}_{\mathrm{KL}}(\tilde{p}_\theta \,||\, p) = \mathbb{E}_{z\sim r(z)}\left[S\big(f_\theta(z)\big) - \log\left|\det \frac{\partial f_\theta(z)}{\partial z}\right|\right] + \text{irrelevant terms}\,. \tag{3.20}$$

In Eq. 3.20, the irrelevant terms that do not depend on the model's parameters are the latent density $\mathbb{E}_{z\sim r(z)} \log r(z)$ and the normalizing factor in the path integral, $\log \mathcal{Z}$.

There is a stark contrast between training the normalizing flow model with this reversed KL and a more traditional approach, such as the one taken in Chapter 2 where we minimised $\hat{D}_{\mathrm{KL}}(p \,||\, \tilde{p}_\theta)$. The difference goes further than exchanging an independent training set for configurations drawn from the model. Some key differences which must be kept in mind are summarised below:

- Training inputs are generated from the model.
- There is no overfitting of the training input but there may be some underfitting and slow convergence. In particular model may get stuck with $\tilde{p}_\theta(\phi) \approx 0$ even if $p(\phi) \gg 0$.
- Seeing as there is no dataset, a minibatch or batch of data is simply a set of states which is used for a single update of the model parameters. We tend to avoid using the term epochs or training iterations and instead quantify training length by the total number of states generated during the training.

We have actually sidestepped several of the major difficulties normally faced during training. In particular, since each batch of training inputs is stochastically generated on-demand, and never recycled, 'over-fitting' of training data is not an issue. The problem that we are most likely to encounter is one of insufficient

flexibility to resolve all of the features in the target density, leading to a model which *under-fits* the target.

Something which should be re-emphasised is the independence of configurations generated from the model. Since the latent variables are generated completely independently from one another, the candidate field configurations are also i.i.d. This, again, contrasts samples generated from the RBM which were produced using Gibbs sampling which induced some correlation between successive configurations. This is an attractive feature of any generative model which uses a latent variable in this manner such as normalizing flows, variational auto-encoders and GANs.

3.3.2 Building flexible models

When considering potential transformations for the layers g_i, there are two potentially conflicting requirements that will need to be met with a compromise. Firstly, the flow will need to be highly flexible in order to start with uncorrelated Gaussian variables and distil the complex features of a system near to criticality, which will include non-trivial correlations on multiple scales. On the other hand, the Jacobian determinant in Eq. 3.16 must be tractable since we are still required to evaluate $\tilde{p}_\theta(\phi)$ in order to perform the reweighting that guarantees convergence to the true target. Furthermore, the speed at which models can be trained and sampled from will depend on the efficiency with which the Jacobian determinant can be computed. Clearly this constraint is at odds with the goal of using invertible transformations to map simple distributions to complex ones. It is very challenging to define sufficiently expressive transformations without rendering the Jacobian term intractable.

The key development that facilitated normalizing flows to become competitive with more flexible DGMs at performing benchmark tasks (such as image synthesis) was a particular type of highly flexible element-wise transformation, now known as a *coupling layer* [99, 100, 114]. The coupling layer construction involves splitting the inputs into two groups. For ϕ^4 theory with a single degree of freedom at each lattice site, this equates to partitioning the lattice into Λ^A and Λ^P, which we refer to as the 'active' and 'passive' partitions, respectively. Defining $v_1 \equiv z$ and $v_{I+1} \equiv \phi$, where I is the number of coupling layers, the transformation of the

coupling layer on each partition is defined by

$$g_i \;:\; \begin{cases} v_i^P \mapsto v_i^P \\ v_i^A \mapsto C_i\big(v_i^A; \mathbf{N}_i(v_i^P)\big) \end{cases} \quad , \tag{3.21}$$

with $v_i^P, v_i^A \in \mathbb{R}^{|\Lambda^A|}$ being a set of variables built out of elements $\{v_{i,x} \,|\, x \in \Lambda^A\}$. The function C_i transforms the active partition and depends on a set of parameters $\mathbf{N}_i(v_i^P)$ that are themselves functions of the passive partition. In the examples that we will consider here, these parameters are the output of fully-connected feed-forward neural networks.

Throughout this Chapter, neural network outputs, exclusively, will be denoted by bold letters, and it will be left to the presence or absence of indices (e.g. i for the layer index, x for the lattice sites) to specify the cardinality of sets.

In block notation, the Jacobian for a coupling layer is

$$\frac{\partial g_i}{\partial v_i} = \begin{pmatrix} \mathbb{I} & 0 \\ \frac{\partial C_i}{\partial v_i^P} & \frac{\partial C_i}{\partial v_i^A} \end{pmatrix} . \tag{3.22}$$

Since this matrix is triangular, the determinant is simply the product of terms on the diagonal

$$\det \frac{\partial g_i}{\partial v_i} = \prod_{x \in \Lambda^A} \frac{\partial C_{i,x}}{\partial v_{i,x}} . \tag{3.23}$$

By alternating the active A and passive P partitions after each coupling layer, and composing the flow model of at least three layers, we ensure that every lattice site is updated using information from every other one. Thus, the coupling layers allow us to model correlated target densities using uncorrelated latent variables, crucially without additional expense in the computation of the Jacobian determinant.

Since $\det AB = \det A \det B$, a sequence of I coupling layers induce the following Jacobian determinant,

$$\log \left| \det \frac{\partial f_\theta}{\partial z} \right| = \sum_{i=1}^{I} \log \left| \det \frac{\partial g_i}{\partial v_i} \right| . \tag{3.24}$$

As a practical consideration when writing the code for a normalizing flow, these terms should be accumulated alongside the transformation of the field variables so that only a single pass through all the coupling layers yields both the transformed

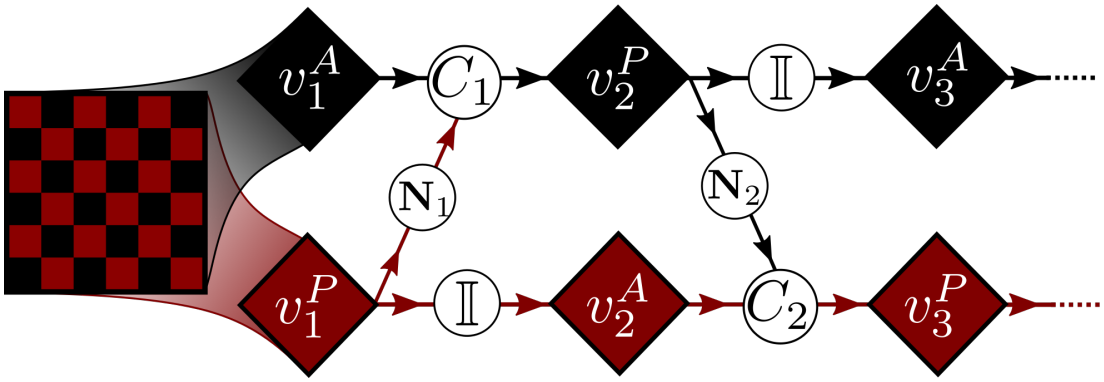

Figure 3.1 *Graphical representation of the first coupling block under the checkerboard partitioning scheme. The latent Gaussian variables, $v_1 \equiv z$, are split into two partitions (the red and black nodes). The i-th coupling layer transforms the active partition $C_i : v_i^A \mapsto v_{i+1}^P$ using information from the passive partition, v_i^P, via the neural network(s) $\mathbf{N}_i(v_i^P)$. $\mathbb{I}$ denotes the identity transformation. Note that there is no need to concatenate the active and passive partitions until after the final coupling layer.*

variables and the Jacobian determinant in Eq. 3.24 for each configuration being generated. This avoids unnecessary duplication of work during the training.

The choice of partitioning is not expected to make a huge difference on the success of the model since, as previously mentioned, with more than 3 coupling layers correlations can occur between any of the output field variables. we will implement a *checkerboard* partitioning featuring in Fig. 3.1, which ensures that each lattice site is directly influenced by its closest neighbours. This partitioning was adopted in [1], and seems sensible considering the interactions in our theory. We will often refer to a pair of coupling layers, which together transform every degree of freedom once, as a *coupling block*. From hereon, we will drop the A and P superscripts and assume we are always talking about transforming a set of variables v_i belonging to the active partition. Furthermore, we will denote the neural networks without an explicit dependence on the passive partition.

3.3.3 Affine and additive transformations

Affine coupling layers were introduced by [100] as part of the *Real NVP* architecture. The element-wise transformation multiplies and shifts each degree of freedom, and is commonly written in vector form,

$$C_i^{\text{aff}}(v_i\,;\,\mathbf{s}_i,\mathbf{t}_i) = (v_i - \mathbf{t}_i)\odot e^{-\mathbf{s}_i}\,, \tag{3.25}$$

where $\mathbf{s}_i$ and $\mathbf{t}_i$ are modelled by neural networks with $|\Lambda^A|$ outputs, and $\odot$ is the element-wise product. An interesting feature of this transformation is that it can be inverted *without the need to invert the neural networks.*

Using Eqs. 3.23 and 3.24, a single affine coupling layer contributes

$$\log \left| \det \frac{\partial g_i^{\text{aff}}}{\partial v_i} \right| = - \sum_{x \in \Lambda^A} \mathbf{s}_{i,x} \tag{3.26}$$

to the logarithm of the Jacobian determinant. The precursor to *Real NVP* uses volume-preserving additive coupling layers [99], such that Eq. 3.25 reduces to the shift by $\mathbf{t}_i$ only,

$$C_i^{\text{add}}(v_i \,;\, \mathbf{t}_i) = v_i - \mathbf{t}_i \,. \tag{3.27}$$

In our implementation of these coupling layers, we standardise the inputs to the neural networks such that they have unit variance, and do not apply activation functions to the output layer of these neural networks. We also append a global rescaling transformation after all of the coupling layers have acted, which can have a learnable scale parameter.

As an inexpensive yet remarkably expressive flow architecture, Real NVP has achieved widespread success and is frequently taken as a benchmark model to which new flow models are compared. However, more sophisticated flows using more flexible transformations have since achieved superior results on a number of standard datasets — see e.g. Refs. [114–122]. This motivates us to explore one of the prominent alternatives.

3.3.4 Rational quadratic splines

Splines are functions defined piecewise by polynomials. Coupling layers using spline-based transformations were introduced in [118] and further developed in [120, 121]. We will focus on the most flexible member of the family, based on a continuously differentiable interpolant presented in [123], in which the polynomials are rational quadratics [121].

A rational quadratic spline (RQS) transformation $C_{i,x}^{\text{rqs}}$ is defined for a single degree of freedom by K rational quadratics, referred to as the *segments* of the spline. These segments are joined end-to-end at a set of *knots* $\left\{ \left(v_{i,x}^k, C_{i,x}^{\text{rqs}}(v_{i,x}^k) \right) \,|\, k = 0, \ldots, K \right\}$ such that the result is a strictly monotonic, C^1-differentiable function on the

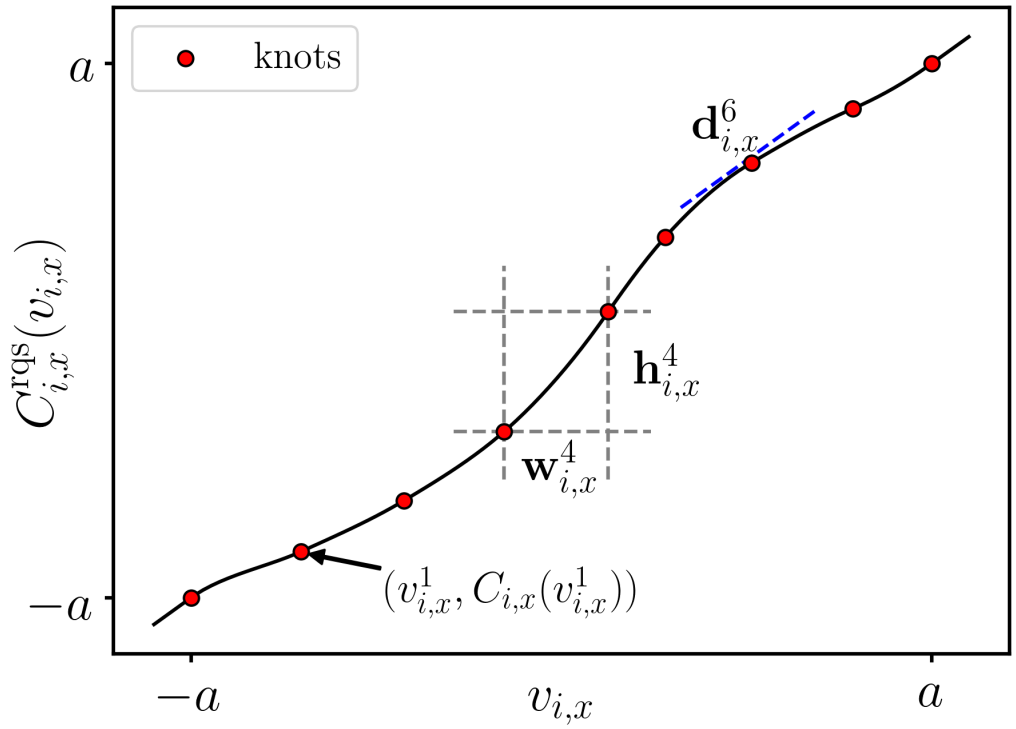

Figure 3.2 *An example 8-segment rational quadratic spline transforming the degree of freedom at lattice site x. $\mathbf{w}^k_{i,x}$ and $\mathbf{h}^k_{i,x}$ are the widths and heights of the rectangle containing the k-th polynomial segment. $\mathbf{d}^k_{i,x}$ is the derivative at the k-th knot.*

interval $[-a, a]$, which will be chosen in order to contain essentially all of the probability mass.

Given a reference point, the parametrization provided by [123] requires $3K+1$ strictly positive parameters to uniquely specify this function: the side lengths $(\mathbf{w}^k_{i,x}, \mathbf{h}^k_{i,x})$ of the K rectangles which have adjacent knots on their opposing corners, *i.e.* their widths and heights, and the derivatives $\mathbf{d}^k_{i,x}$ at the $K+1$ knots. Note that our choice of labelling, featuring in Fig. 3.2, means that the endpoints of the k-th segment are the $(k-1)$-th and k-th knots. This gives us a set of $|\Lambda^A| \times (3K+1)$ parameters for the coupling layer,

$$\mathbf{N}_i = \left\{ \mathbf{w}^k_{i,x}, \mathbf{h}^k_{i,x}, \mathbf{d}^k_{i,x}, , \mathbf{d}^0_{i,x} \,|\, k = 1, \ldots, K;\ x \in \Lambda^A \right\}. \tag{3.28}$$

For later convenience, define the slopes of the straight lines connecting adjacent knots as

$$\frac{C^{\mathrm{rqs}}_i(v^k_i) - C^{\mathrm{rqs}}_i(v^{k-1}_i)}{v^k_i - v^{k-1}_i} = \frac{\mathbf{h}^k_i}{\mathbf{w}^k_i} \equiv \mathbf{s}^k_i \geq 0\,, \tag{3.29}$$

and re-express the variables being transformed as

$$\frac{v_i - v_i^{\ell-1}}{\mathbf{w}_i^\ell} \equiv \alpha_i \in [0,1]^{|\Lambda^A|}\,, \tag{3.30}$$

which defines the fractional positions of these variables within the specific segment in which they are located, whose index we label by $k = \ell$. Note that each degree of freedom $v_{i,x}$ must first be sorted into the appropriate segment (i.e. the value of ℓ determined) using e.g. bisection search, which is not too expensive since the knots are already sorted into ascending order.

Using the 0th knot at $\big(v_{i,x}^0, C_{i,x}^{\rm rqs}(v_{i,x}^0)\big) = (-a,-a)$ as the reference point, the RQS transformation and its gradient can then be written, for each degree of freedom, using Eqs. 3.31 and 3.32.

$$C_{i,x}^{\rm rqs}(v_{i,x}\,;\,\mathbf{N}_{i,x}) = -a + \sum_{k=1}^{\ell-1}\mathbf{h}_{i,x}^k + \frac{\mathbf{h}_{i,x}^\ell\big[\mathbf{s}_{i,x}^\ell\alpha_{i,x}^2 + \mathbf{d}_{i,x}^{\ell-1}\alpha_{i,x}(1-\alpha_{i,x})\big]}{\mathbf{s}_{i,x}^\ell + (\mathbf{d}_{i,x}^{\ell-1} + \mathbf{d}_{i,x}^\ell - 2\mathbf{s}_{i,x}^\ell)\alpha_{i,x}(1-\alpha_{i,x})}\,, \tag{3.31}$$

$$\frac{1}{\mathbf{w}_{i,x}^\ell}\frac{dC_{i,x}^{\rm rqs}}{d\alpha_{i,x}} = \frac{(\mathbf{s}_{i,x}^\ell)^2\big[\mathbf{d}_{i,x}^\ell\alpha_{i,x}^2 + 2\mathbf{s}_{i,x}^\ell\alpha_{i,x}(1-\alpha_{i,x}) + \mathbf{d}_{i,x}^{\ell-1}(1-\alpha_{i,x})^2\big]}{\big[\mathbf{s}_{i,x}^\ell + (\mathbf{d}_{i,x}^{\ell-1} + \mathbf{d}_{i,x}^\ell - 2\mathbf{s}_{i,x}^\ell)\alpha_{i,x}(1-\alpha_{i,x})\big]^2}\,. \tag{3.32}$$

Taking the logarithm of Eq. 3.32 and summing over all x in the active partition yields the contribution to the logarithm of the Jacobian determinant from one RQS coupling layer.

The advantage of using this parametrization should now be clear; all that is required to guarantee that the transformation is *strictly monotonic* (and therefore invertible) is that every parameter in $\mathbf{N}_i$ is strictly positive. It is also particularly simple to enforce the desired normalization:

$$\sum_{k=1}^{K}\mathbf{w}_{i,x}^k = \sum_{k=1}^{K}\mathbf{h}_{i,x}^k = 2a\,. \tag{3.33}$$

The function defined by Eq. 3.31 is an interpolant for the set of knots, so the problem of representing complicated transformations reduces to one of generating a sufficient number of knots with sufficient accuracy.

After fixing $\mathbf{d}_i^0 = \mathbf{d}_i^K = \{1\}^{|\Lambda^A|}$, we let a *single* neural network generate the remaining $|\Lambda^A| \times (3K-1)$ parameters in the RQS layer, using the $|\Lambda^P|$ field variables in the passive partition as inputs. We take the unconstrained outputs of

the neural net — denoted below with a hat — and split them into widths, heights, and derivatives. Positivity of the $\mathbf{h}^k_{i,x}$ and $\mathbf{w}^k_{i,x}$, as well as the correct normalization, is enforced by passing the unconstrained widths and heights through a *softmax* activation function

$$\mathbf{h}^k_{i,x} = \frac{e^{\widehat{\mathbf{h}}^k_{i,x}}}{\sum_{k'=1}^{K} e^{\widehat{\mathbf{h}}^{k'}_{i,x}}} \times 2a\,, \tag{3.34}$$

which is essentially the generalisation of the logistic function to multiple variables. The derivatives are instead passed through a *softplus* activation function

$$\mathbf{d}^k_{i,x} = \log(1 + e^{\widehat{\mathbf{d}}^k_{i,x}})\,, \tag{3.35}$$

which guarantees their positivity.

To ensure that the variables which enter the spline transformation fall within the interval $[-a, a]$, we generally chose $a = 5$ and standardised the inputs before the first RQS layer by dividing them by the standard deviation, taken over both the batch and the lattice sites. To catch the edge cases of input variables falling outside of this interval, we extended the definition of the transformation to be the identity outside of $[-a, a]$, while fixing the derivatives at the external knot points to be unity to ensure that the transformation remains everywhere differentiable.

Neural networks We emphasise that the parameters for coupling layers are the outputs of neural networks acting on the passive partition, for example

$$\mathbf{N}_x(v^P) = \sum_{j=1}^{H} w^2_{xj}\, \sigma\Big(\sum_{x' \in \Lambda^P} w^1_{jx'} v^P_{x'} + b^1_j \Big) + b^2_x\,, \tag{3.36}$$

which represents feed-forward neural networks with a single hidden layer containing H elements and a non-linear activation function σ [124]. Where, as usual, w is a matrix containing the network weights, and b is a vector of biases.

3.3.5 Enforcing sign-reversal equivariance

ϕ^4 theory possesses a $\mathbb{Z}_2$ symmetry corresponding to invariance of the action under a global sign-reversal of the field, implying

$$p(-\phi) = p(\phi)\,. \tag{3.37}$$

Since the latent variable already possesses $z \mapsto -z$, we can attempt to constrain f_θ in order to preserve this symmetry by construction.

Maps which preserve a symmetry are those which commute with the group action, and are sometimes referred to as being *equivariant* (with respect to the symmetry). It is not difficult to show that in this case the equivariant maps are odd functions, and that equivariance of f_θ requires the coupling layers $g_i(v_i)$ to be equivariant, meaning that the transformations satisfy

$$C_i\big(-v_i^A\,;\,\mathbf{N}_i(-v_i^P)\big) = -C_i\big(v_i^A\,;\,\mathbf{N}_i(v_i^P)\big)\,. \tag{3.38}$$

In this work, $\mathbf{N}_i(v^P)$ are fully-connected feed-forward networks as defined by Eq. 3.36, which are odd functions if we drop the biases and use odd activation functions (e.g. tanh) [90]. If we make these choices for the neural networks in the affine coupling layers, $\mathbf{s}_i$ and $\mathbf{t}_i$, then Eq. 3.38 is trivially satisfied by implementing one additional step, that is to take the absolute value of the output of the $\mathbf{s}_i$ network.

Enforcing equivariance in the RQS transformations is less straightforward because the terms in Eq. 3.31 cannot all be simultaneously odd. We implemented a rather crude workaround that involves splitting the batch of latent variables according to $\operatorname{sgn}\sum_{x\in\Lambda} z_x$ (i.e. the initial 'magnetisation' of each configuration), and treating the two groups slightly differently within the transformation. We transform states with $\operatorname{sgn}\sum_{x\in\Lambda} z_x > 0$, using the RQS transformation in Eq. 3.31. But for states with $\operatorname{sgn}\sum_{x\in\Lambda} z_x < 0$, we reverse the transformation taking $C_i^{\text{rqs}} \mapsto -C_i^{\text{rqs}}$. This is achieved by reversing the order of the network outputs $\mathbf{h}_i$, $\mathbf{w}_i$ and $\mathbf{d}_i$. This approach is not entirely satisfactory, not least because $\partial C_i^{\text{RQS}}/\partial v_i$ varies discontinuously as $\sum_{x\in\Lambda} z_x$ changes sign, which may introduce instability into the training. More importantly, this has broken the diffeomorphism property of the normalizing flows and so it could have the severe impact of breaking ergodicity.

It is worth bearing in mind that enforcing symmetries is not absolutely necessary; the Metropolis-Hastings algorithm is guaranteed to converge to the correct target, and therefore reproduce all of its symmetries, as long as the transition kernel is ergodic. We remind the reader that a sufficient condition is $\tilde{p}_\theta(\phi) > 0\ \forall \phi \in \mathbb{R}^{|\Lambda|}$ [104] (so that every configuration has a finite probability of being generated), and that this is guaranteed (for a sensible choice of $r(z)$) since f_θ is a bijection. Nevertheless, a guiding principle of optimisation is that it is generally more efficient to enforce known constraints by construction, and benefits of doing so for

normalizing flows have been reported in Refs. [89, 91, 125].

3.4 Related work

The first demonstration of a normalizing flow forming the basis of a sampling algorithm for lattice field theory was provided in [1] for two-dimensional ϕ^4 theory, using the *Real NVP* architecture described in Sec. 3.3.3. In this study we validate our results by directly comparing the observables obtained from our models, to the results quoted in that study. Still with ϕ^4 as the target theory, in [90] they used an even more bare-bones flow where the coupling layers simply shift the field variables, in such a way that the symmetry under $\phi \to -\phi$ is preserved. We pivot in the opposite direction with respect to Ref. [90] by using coupling layers that are *more* flexible than those in *Real NVP*.

More recent progress has been on the side of developing the necessary machinery to apply these ideas to lattice *gauge* theories; specifically, those that are invariant under local U(N) or SU(N) transformations. In Ref. [107], they explored several possible approaches to using normalizing flows in cases where the field variables are defined on an n-sphere or n-torus. A procedure for constructing normalizing flows that are equivariant under gauge transformations was initially developed in [89] for the U(1) case and then extended to SU(2) and SU(3) in [91]. By definition, a gauge-equivariant flow is one that commutes with the action of the gauge group, which implies that gauge *in*variance is preserved by the flow. Hence, representative samples of gauge fields can be generated using latent variables drawn from the uniform (Haar) measure for the gauge group, and passing them through a gauge-equivariant flow. A code-based introduction to these methods was very recently provided [92].

Another recent and highly relevant study showed that, in certain cases at least, it is possible to use a normalizing flow to sample from a theory possessing a 'sign problem', which is to say the action is complex and $\exp(-S)$ cannot be interpreted as a measure of probability [93].

Several alternative ideas that involve training parametric models to perform collective updates predate the use of normalizing flows. For example, in the 'self-learning Monte Carlo' method [126] the parametric model describes an effective action for a spin system with n-th nearest neighbour interactions whose couplings

have been inferred from pre-generated training data, which can then be used to generate Wolff cluster updates [75]. Some studies have taken RBMs, discussed in Chapter 2, and embedded them in traditional MCMC algorithms [83, 84]. Although as we already mentioned, training the RBM has so far relied on training from pre-generated configurations and sampling from the RBM introduces its own autocorrelation.

Whilst Generative Adversarial Networks (GANs) can be more flexible than normalizing flows, since $\tilde{p}_\theta(\phi)$ is defined implicitly and cannot be directly computed, they require a lot of additional machinery on top of the GAN itself to ensure that the distribution being sampled from is close to the correct one [85, 87], or otherwise quantify the discrepancy [88].

While there have been substantial advances in the use of machine learning to extract physical information for lattice field theories, the generation of samples from some approximation of the true path integral has generally come as an add-on when the tool being used is a DGM. In contrast, the key strength of normalizing flows is the explicit and tractable density $\tilde{p}_\theta(\phi)$ which makes exact sampling much easier than some of the aforementioned tools, using reweighting or the Metropolis test.

3.5 Experimental setup

3.5.1 Field theory and observables

For the main part of our study we used the following action,

$$S(\phi) = \sum_{x\in\Lambda} \left[-\beta \sum_{\mu=1}^{2} \phi_{x+e_\mu}\phi_x + \phi_x^2 + \lambda(\phi_x^2 - 1)^2 \right], \tag{3.39}$$

which describes a discretised analogue of two-dimensional scalar ϕ^4 theory with dimensionless couplings β and λ, defined on a periodic lattice Λ, and using e_μ to denote a unit lattice vector in the μ-th dimension. Experiments with the non-interacting theory used the 'standard' action described in Appendix B.1, which is given by Equation (B.6) with $g_0 = 0$. We focus on isotropic lattices with $6^2 \leq |\Lambda| \leq 20^2$ sites.

A nice feature of the parametrization given above is that the limit $\lambda \to \infty$, $\phi^2 \to 1$

is very clearly identified as the Ising model at temperature $T \equiv \beta^{-1}$. Indeed, in the continuum limit ϕ^4 theory belongs to the Ising universality class, with spontaneous breaking of the $\phi \mapsto -\phi$ symmetry occurring along a critical line $\big(\lambda, T_c(\lambda)\big)$ in the space of couplings. Near the phase transition, the dependence of observables on the couplings is typically written in terms of $t = \frac{T - T_c(\lambda)}{T_c(\lambda)}$. For example, the magnetic susceptibility diverges as $\chi \sim t^{-\gamma}$, and the correlation length diverges with a different critical exponent, $\xi \sim t^{-\nu}$. Eliminating t, we see that $\chi \sim \xi^{\gamma/\nu}$.

In a finite volume observables depend on both the couplings and the system size in a non-trivial manner, and their behaviour in the critical region is described by finite-size scaling. For example, the susceptibility in a volume of linear extent L can be written in the following manner,

$$\chi = \xi^{\gamma/\nu} g_\chi(L/\xi) \tag{3.40}$$

in which finite-volume effects have been bundled into a dimensionless scaling function $g_\chi(L/\xi)$, which we notice must tend towards a constant value in the thermodynamical limit, and approach $(L/\xi)^{\gamma/\nu}$ for $L \ll \xi$ so as to act as a cutoff.

We now need to specify how we actually measure observables on the lattice. The basic building block is the two point correlation function,

$$G(y) = \frac{1}{|\Lambda|} \sum_{x \in \Lambda} \Big\langle \big(\phi_{x+y} - \langle\phi\rangle\big)\big(\phi_x - \langle\phi\rangle\big) \Big\rangle \,, \tag{3.41}$$

and its Fourier transform,

$$\tilde{G}(q) = \sum_{y \in \Lambda} e^{iq\cdot y} G(y) \,. \tag{3.42}$$

We have used the translation invariance of (3.39) to take a volume-average in (3.41) for the simple reason that it improves the statistics.

The susceptibility is identified with $\tilde{G}(0)$, but in the classical spin setting it is often expressed in terms of the magnetisation $M(\phi) = \sum_{x \in \Lambda} \phi_x$,

$$\tilde{G}(0) \equiv \chi = \frac{1}{|\Lambda|} \Big\langle \big(M - \langle M\rangle\big)^2 \Big\rangle \,, \tag{3.43}$$

Estimators for these observables are easily obtained by exchanging $\langle . \rangle$ for a

sample mean, and uncertainties estimated using the bootstrap method [127, 128]. However, without explicitly breaking the $\mathbb{Z}_2$ symmetry one will always measure $\langle\phi\rangle = |\Lambda|^{-1}\langle M\rangle = 0$, so if one is interested in the phase transition one can compute separate sample averages for configurations with positive and negative magnetisation, to properly account for the fact that the field variable distribution is bimodal.

The correlation length requires a little more work to measure. It is the longest mode in the spectrum of $\sum_{x_1=0}^{L-1} G(x_1, x_2)$, the correlation between one-dimensional 'slices'. For sufficiently large separations, x_2, this takes the form of a pure exponential (a cosh due to lattice periodicity),

$$\sum_{x_1=0}^{L-1} G(x_1, x_2) \equiv \hat{G}(x_2) \propto \cosh\left(\frac{x_2 - L/2}{\xi}\right), \tag{3.44}$$

from which the correlation length can be extracted through a fit or by computing

$$\xi^{-1}(x_2) = \operatorname{arcosh}\left(\frac{\hat{G}(x_2+1) + \hat{G}(x_2-1)}{2\hat{G}(x_2)}\right). \tag{3.45}$$

In general, this can be challenging due to low signal/noise ratio at large separations, but with $L \leq 20$ we also suffer from having very few data points to fit. To slightly improve the situation, we average over the two dimensions when computing (3.45).

Another option exploits the fact that the lattice propagator takes the form $\tilde{G}(q) \propto \left(\sum_\mu 4\sin^2(q_\mu/2) + \xi^{-2}\right)^{-1}$ in the low-momentum limit, which lets us write [129]

$$\xi^2 = \frac{1}{2}\sum_{\mu=1}^{2} \frac{1}{4\sin^2(\pi/L)}\left(\frac{\tilde{G}(0)}{\operatorname{Re}\tilde{G}(\hat{q}_\mu)} - 1\right). \tag{3.46}$$

Here, $\hat{q}_1 = (2\pi/L, 0)$ and $\hat{q}_2 = (0, 2\pi/L)$ are the smallest possible non-zero momenta, and we have used $\tilde{G}(q) + \tilde{G}(-q) = 2\operatorname{Re}\tilde{G}(q)$.

Our intention will be to tune the couplings so as to obtain systems with correlation length $\xi = L/4$, meaning that as we increase the lattice size we are studying essentially the same theory with an increasingly fine resolution. The reason for this is two-fold: firstly this proportionality was used in [1] so if we wish to compare results we should compare like with like. From the physical perspective we only see the effect of the number of degrees of freedom on algorithmic efficiency, as we keep the physical size in units of the correlation length constant. The choice proportionality constant (four) is a reasonable trade-off between the rate at which

L	λ	β	ξ (fit)	ξ from Eq. 3.45	ξ from Eq. 3.46
6	0.5	0.537	1.57(2)	1.525(3)	1.501(3)
8	0.5	0.576	2.05(5)	2.005(2)	1.990(2)
10	0.5	0.601	2.53(2)	2.529(2)	2.524(3)
12	0.5	0.616	3.10(8)	3.013(3)	3.010(5)
14	0.5	0.626	3.40(4)	3.471(5)	3.487(8)
16	0.5	0.634	4.03(9)	3.940(3)	3.970(4)
18	0.5	0.641	4.56(5)	4.502(6)	4.555(9)
20	0.5	0.645	5.1(2)	4.903(9)	4.96(1)

Table 3.1 *ϕ^4 couplings and correlation length measurements for the main part of our study. The inverse temperature β was tuned such that $\xi \approx L/4$ for each lattice size.*

criticality is approached as we increase L, and the size of finite-volume effects contained within scaling functions. Fixing $\lambda = 0.5$ and allowing β to vary, we obtained three separate predictions for the value of ξ showing that these values of β correspond to $\xi = L/4$ on the symmetric side of the phase transition. These values are provided in Table 3.1. The difference between the different determinations of ξ stem from the different assumptions that go into the estimations. The largest discrepancies in the table, once the error is taken into account, occur for smaller lattices which makes sense since we would expect there to be more finite size effects and the low momentum approximation is less valid, since the lowest momentum is inversely proportional to the lattice size.

3.5.2 Model details

When investigating the scaling of training costs (Section 3.6.4), we used normalizing flows that are a specific hybrid of affine coupling layers and rational quadratic splines, with the parameters of the transformations generated by fully-connected feed-forward neural networks containing a single hidden layer of size $H = |\Lambda|$. In Sec. 3.6.3 we report on the observations that led us to converge on this particular design.

The metric we use to measure the quality of trained models is the average rate at which configurations generated by the model are accepted when used as proposals for a Metropolis-Hastings simulation. In Sec. 3.6.1 we verify that this acceptance rate entirely governs the integrated autocorrelation times of the resulting Markov chains, as claimed in Sec. 3.2.2, Eq. (3.13).

In Secs. 2.3 and 2.4, we outlined a manual schedule for tuning the learning rate in order to train the RBM. Whilst there is clearly major differences between the two training procedures, there are at least qualitative arguments for reducing the learning rate as the training progresses in order to fine tune the model parameters. We used the ADAM optimisation algorithm [113] to update the parameters of our models. Instead of manually updating the learning rate, we instead anneal it using a cosine schedule,

$$\eta_t = \frac{\eta_0}{2}\left[1 + \cos\left(\frac{t}{T}\pi\right)\right], \tag{3.47}$$

where T is the total number of training iterations. Note that this learning schedule requires that we specify T before training begins and that there is no additional stopping criteria (such as the loss function plateauing). We use the ADAMW variant [130] of the optimiser along with warm restarts [131]. Warm restarts amount to restarting the training, resetting $t = 0$ in Eq. 3.47 which allows us to continue training if we are not happy with the outcome after T iterations. After some experimentation with faster initial learning rates, which typically resulted in lower acceptances if the number of training iterations was large, we generally opted for $\eta_0 = 0.001$.

The batch size, i.e. the number of configurations used to estimate the objective function at each training iteration, varied from 250 to 32000 configurations. In the vast majority of cases the difference between the batch size and the number of training iterations was a factor of one, two or four. Note that these batch sizes are much larger than those conventionally used in stochastic optimisation. In fact, it is quite typical to intentionally aim for a highly stochastic trajectory through the space of parameters, by using a very small number of training inputs (as low as 2 in Reference [132]) for each update of the model's parameters. This may seem surprising, particularly since the graphical processing units (GPUs) on which these models are run are entirely optimised for highly parallel computations, so a small batch size is an under-utilisation of these capabilities. The motivations behind this choice are that the stochasticity reduces the tendency of the model to over-fit the training inputs or otherwise get stuck in local optima [132, 133], and instead are more likely to find the global minimum [134]. However, we have no reason to prefer small batch sizes a priori; as explained in Sec. 3.3.1, the problem of over-fitting training inputs does not apply to us, and we expect the issue of local optima to be alleviated, to some extent, thanks to stochasticity inherited from the random number generator that produces our training inputs. In our

case, the only limitation to the batch size is hardware: at some point we are physically unable to increase the size of the batch without significantly adding to the computational time required to process the batch.

Unless stated otherwise, one can assume the following for all models presented in the remainder of this paper:

- The ϕ^4 couplings are given by Table 3.1.
- The flow comprises a number of affine coupling blocks followed by a single rational quadratic spline coupling block.
- $\mathbb{Z}_2$ equivariance is enforced in the affine and additive coupling layers, as described in Section 3.3.5.
- The splines have 8 segments and do not have $\mathbb{Z}_2$ equivariance enforced.
- Neural networks have a single hidden layer containing exactly $|\Lambda|$ (i.e. L^2) elements, as defined in Equation (3.36) with $H = |\Lambda|$.
- We do not apply an activation function to the output layer of the $\mathbf{s}$ and $\mathbf{t}$ networks in the affine (or additive) layers.
- Metropolis-Hastings simulations ran for 10^5 steps.
- In figures, data points and error bars are an average and range taken over three identical models with different random initialisations.

A significant amount of effort was made to release our code, ANVIL [135], for public use. The code uses the PyTorch library [136] for constructing and training models, and Reportengine [137], a declarative framework for performing scientific analysis. Models are specified by human readable runcards which allows for rapid experimentation with different flow architectures and hyperparameters.

3.5.3 Summary of the procedure

A training iteration consists of the following steps:

1. Sample from Eq. (3.17) to generate a batch of N latent configurations, $(z^{(1)}, z^{(2)}, \ldots, z^{(N)})$ where $z^{(n)} \sim r(z)$. Each configuration $z^{(n)}$ comprising $|\Lambda|$ uncorrelated Gaussian variables.

2. Pass these variables through the layers of the model, calculating the logarithm of the Jacobian determinant, $\log|\det \partial g_i/\partial v_i|$, for each layer g_i as it transforms one of the two partitions. This results in N candidate field configurations and N Jacobian determinants $\log|\det \partial f_\theta/\partial z|$ corresponding to the full transformation $\phi = f_\theta(z)$.

3. Compute the action, Eq. (3.39), for the batch of candidate field configurations.

4. Average the action and Jacobian over the batch, to provide an estimate of the reverse Kullbach-Leibler divergence, Eq. (3.20). We rely on PyTorch's 'autograd' machinery [136] in order to propagate the gradients of loss with respect to the neural network parameters using the backpropagation algorithm [138].

5. Update the parameters of the model by a small increment in the direction of steepest gradient using the ADAM or ADAMW optimisation algorithm.

Once we have a trained model, we move onto the sampling. We generate a large sample of candidate configurations from the model, along with their Jacobian terms, and immediately calculate the quantity $\log w(\phi) = -\log \tilde{p}_\theta(\phi) - S(\phi)$ for each candidate configuration. We are now fully equipped to run a Metropolis-Hastings simulation as described in Sec. 3.2. For the Metropolis test we simply exponentiate $\log w(\phi') - \log w(\phi)$ to obtain the acceptance probability $A(\phi \to \phi')$, see Eq. (3.9) with $q(\phi \mid \phi') = \tilde{p}_\theta(\phi)$ and $p(\phi) \propto \exp(-S(\phi)))$.

3.6 Results

3.6.1 Proof of principle

As an initial test, we fixed $\lambda = 0.5$ and trained the affine-spline models whilst varying the inverse temperature β in order to cross the phase transition. Fig. 3.3 shows that high acceptance rates are possible in both the symmetric and the broken phase of ϕ^4 when using a unimodal Gaussian prior. As was expected, the model becomes increasingly challenging to train as the phase transition is approached. Note that Fig. 3.3 does not highlight that the training is probably easier for short correlation lengths, in part because the fully-connected neural

networks will contain a high level of redundancy since many degrees are effectively decoupled.

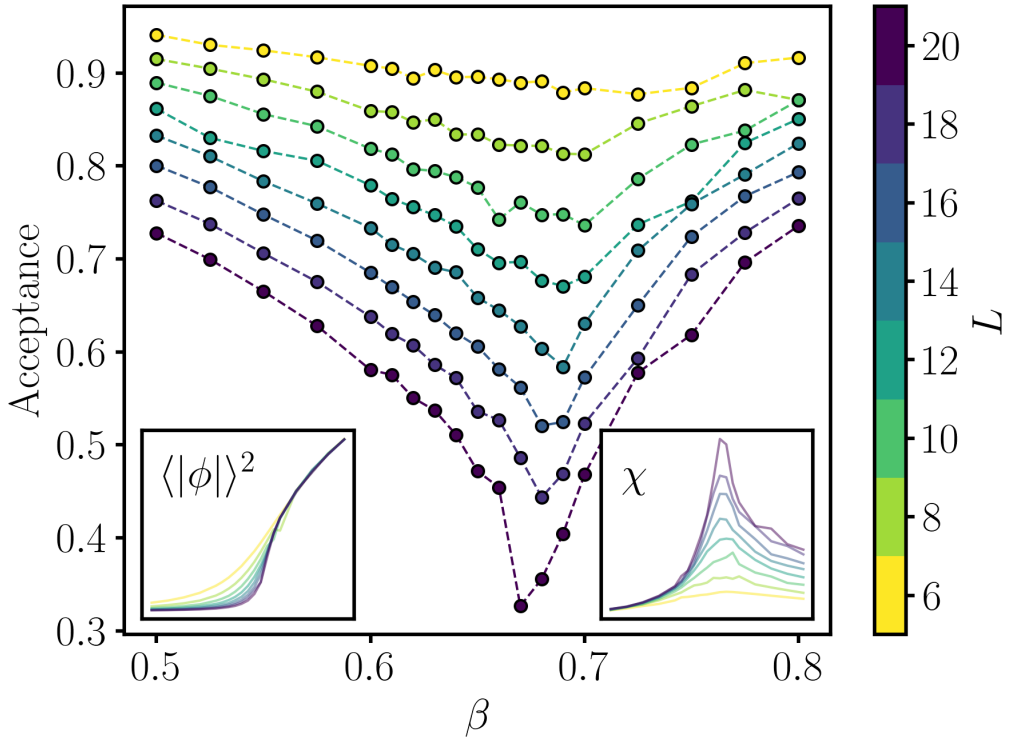

Figure 3.3 *Metropolis-Hastings acceptance rates for a set of models trained at different values of the inverse temperature, β, crossing the critical temperature ($\beta \approx 0.67$ for largest lattice). The inset figures show the magnetisation and susceptibility over the same range of β. Models consisted of a block of affine layers followed by a spline block. The affine layers had $\mathbb{Z}_2$ equivariance enforced as described in Sec. 3.3.5, but this was only true of the spline layers in the low-temperature phase (see Fig. 3.11 for explanation). Emphasis was placed on like-for-like comparison (rather than maximising the acceptance rate); for each lattice size, models were identical and were trained in an identical fashion for a fixed number of iterations.*

Something which made this kind of scan, across parameter β, more efficient was using a single model that is initially trained at a high temperature. Then, by adjusting the temperature over a sequence of training phases, the training doesn't start from scratch. Instead, a model trained at temperature β_0 can be re-trained at temperature $\beta_0 + \delta\beta$ with less effort. Note the similarity between this approach and the annealing which was used in Sec. 2.2.2 to estimate the partition function of the RBM, the key difference being that here we are training a new model starting from a pre-existing one trained on a similar theory.

3.6.2 Acceptance rates and autocorrelation times

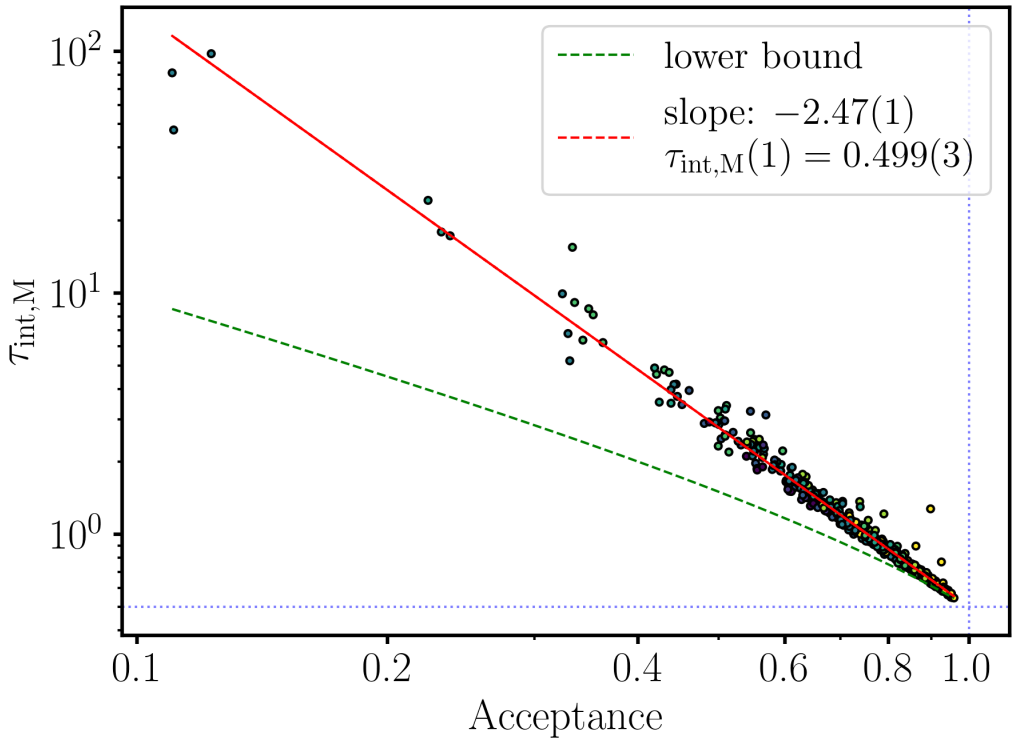

Figure 3.4 *Relationship between Metropolis-Hastings acceptance fraction and integrated autocorrelation time. Both the gradient and the intercept were left unconstrained in the least-squares fit. The theoretical lower bound from Equation* (3.14) *is also plotted in green, for comparison.*

In order to test the claim in Sec. 3.2.2, that when the proposal distribution $q(\phi' \mid \phi)$ in the Metropolis test is not conditioned on the current state (in other words all candidate fields are independent) the autocorrelation is entirely dependent on the acceptance rate of the proposals Eq. 3.13, we perform some empirical tests. Since this has nothing to do with the specifics of the flow model, the system size or the values of the couplings, we simply combined results from a large number of previously trained models (539) to verify this property.

Fig. 3.4 shows a tight relationship between the integrated autocorrelation time τ_{int} and the acceptance, which supports this claim. The geometric lower bound on the integrated autocorrelation time from Eq. (3.14) is also plotted, but we find that the relationship between acceptance rate and integrated autocorrelation time

is fit rather well by a power law,

$$\tau_{\text{int}} \approx \frac{1}{2}\,\mathbb{E}_{\phi,\phi'\sim\tilde{p}_\theta}\left[A(\phi\to\phi')\right]^{-2.48(1)}. \tag{3.48}$$

Table 3.2 contains no new information, but rephrases this power-law relation in terms of the acceptance rate required for the effective sample size to be a particular fraction of the Markov chain length. However, for reasons concerning the measurement of τ_{int}, discussed shortly, we advise against extrapolating to lower acceptances and larger autocorrelation times.

$\frac{N_{\text{eff}}}{\lvert\Phi\rvert}$	1	0.5	0.2	0.1	0.01	0.001
$\mathbb{E}_{\phi,\phi'\sim\tilde{p}_\theta}\left[A(\phi\to\phi')\right]$	1	0.76	0.52	0.40	0.16	0.06

Table 3.2 *Based on the power-law fit from Figure 3.4, $\frac{N_{eff}}{|\Phi|} = (2\tau_{int})^{-1}$ is the ratio between the effective sample size (that controls the statistical error on observables) and the total length of the Markov chain.*

In Fig. 3.4, the autocorrelation time was calculated from autocorrelations in the magnetisation, using a combination of the integrated autocorrelation time from Eq. 3.4 and an estimator of the autocorrelation time. Similarly to Eq. 2.43, we estimate the autocorrelation function as

$$\hat{\Gamma}_{\mathcal{O}}(t) = \left[\frac{1}{N-t}\sum_{i=1}^{N-t}\mathcal{O}(\phi^{(n+t)})\mathcal{O}(\phi^{(n)}) - \bar{\mathcal{O}}^2\right], \tag{3.49}$$

setting the observable $\mathcal{O}$ to be the magnetisation. Then we estimate the integrated autocorrelation time as

$$\hat{\tau}_{\text{int},\mathcal{O}}(W) = \frac{1}{2} + \sum_{t=1}^{W}\frac{\hat{\Gamma}(t)}{\hat{\Gamma}(0)}, \tag{3.50}$$

where W is a finite window which the infinite sum in Eq. 3.4 is truncated to. We demonstrate the dependence of the integrated autocorrelation time of the magnetisation on the window size in Fig. 3.5 and give details of the exact procedure for choosing the optimal window size in App. B.2. We note that although the optimal window size is chosen to minimise the bias and statistical error on the estimate of $\hat{\tau}_{\text{int},\mathcal{O}}(W)$, the estimate is fairly stable for all of the plotted window sizes greater than 6.

We compare the integrated autocorrelation time of the magnetisation to the observable-independent estimator defined in Eq. 3.13 in the left hand plot of

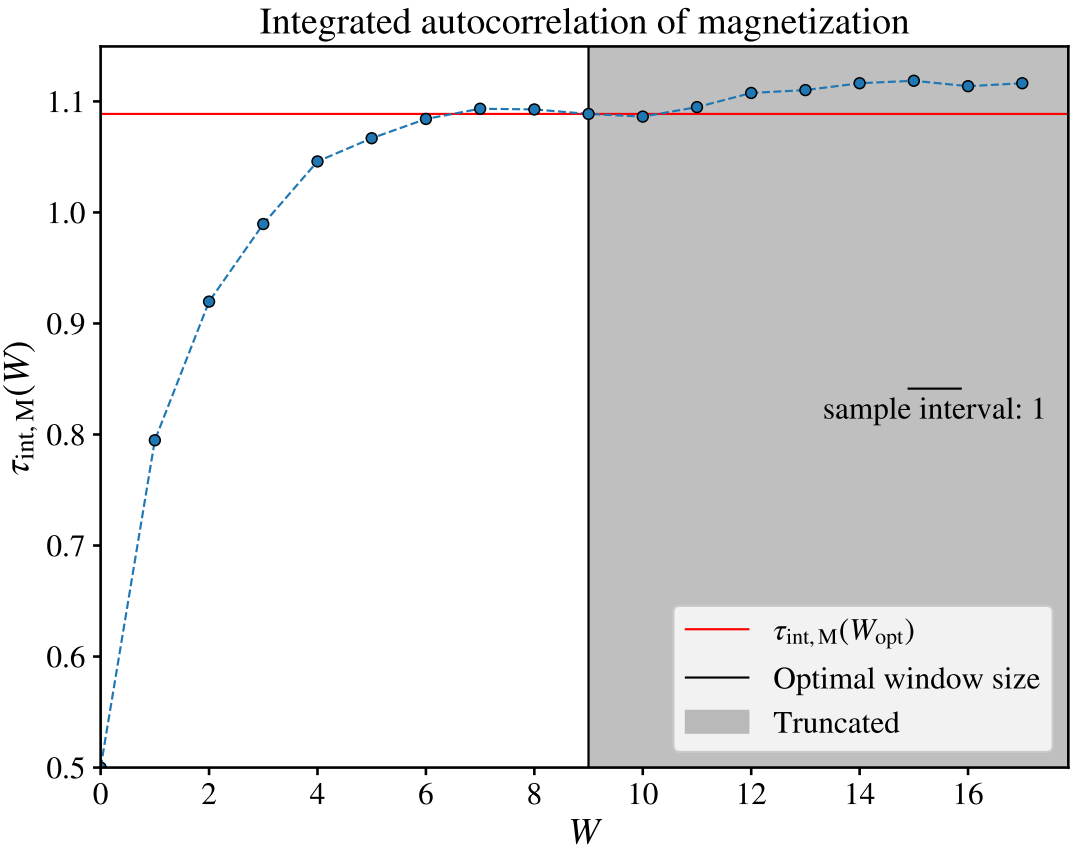

Figure 3.5 *The dependence of the estimate of the integrated autocorrelation time of the magnetisation on the finite window used to estimate it W. For small window sizes the estimate is biased, for larger window sizes the estimate has statistical fluctuations. In practice we choose an optimal window size, using a procedure which is detailed in App. B.2, however we see from the plot that the estimates for plotted window sizes greater than about 6 are fairly consistent with each other.*

Fig. 3.6. There is generally good agreement between the two. Where there is discrepancy between the two, it is systematic: the rejection-based estimator returns a larger integrated autocorrelation time than that calculated using the magnetisation.

In order to understand this phenomenon, we show in the right hand plot of Fig. 3.6 that the discrepancy between $\tau_{\mathrm{int},M}$ and $\tau_{\mathrm{int,\ chain}}$ is directly related to the presence of long periods in the Metropolis-Hastings simulation in which every proposed configuration is rejected. These events can be shown to arise in the tails of the distribution depicted in Fig. 3.7 and are clearly a problem when estimating the autocorrelation time.

The rejection-based estimator is highly sensitive to these long runs of rejections

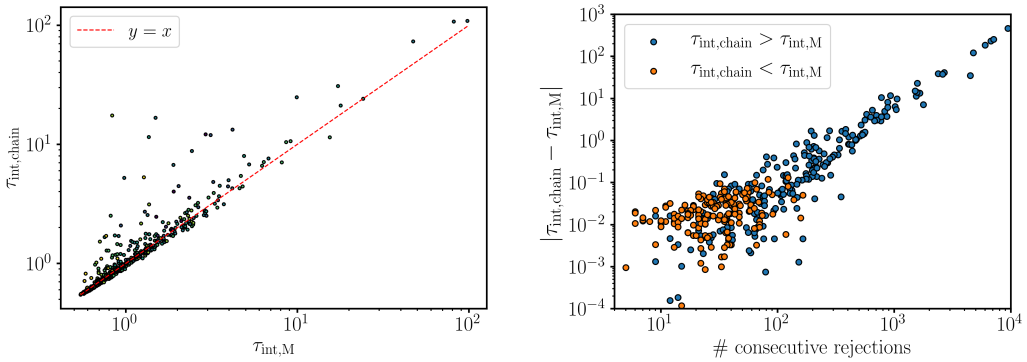

Figure 3.6 *Left: Comparison of alternative estimates of the integrated autocorrelation time. On the x-axis, $\tau_{int,M}$ is calculated in the traditional way, by measuring the autocorrelation function of the magnetisation (see Appendix B.2). On the y-axis, the estimator is based on the Metropolis-Hastings rejection rate, using equations* (3.4) *and* (3.13)*. Right: Discrepancy between the alternative estimates of the integrated autocorrelation time shown on the left, with x-coordinates corresponding to the length of the longest consecutive run of rejections occurring in the sampling phase.*

and hence picks up a large systematic error. Conversely, the traditional estimator is relatively insensitive to a small number of uncharacteristically long periods of consecutive rejections, which may result in an underestimate of the true integrated autocorrelation time.

In Fig. 3.7 we see the distribution describing the length of runs of consecutive rejections become less long-tailed as the acceptance rate increases. Although this trend is somewhat obvious in a qualitative sense, the circumstances by which these long-tailed distributions arise are an interesting facet of the scheme used to train these models.

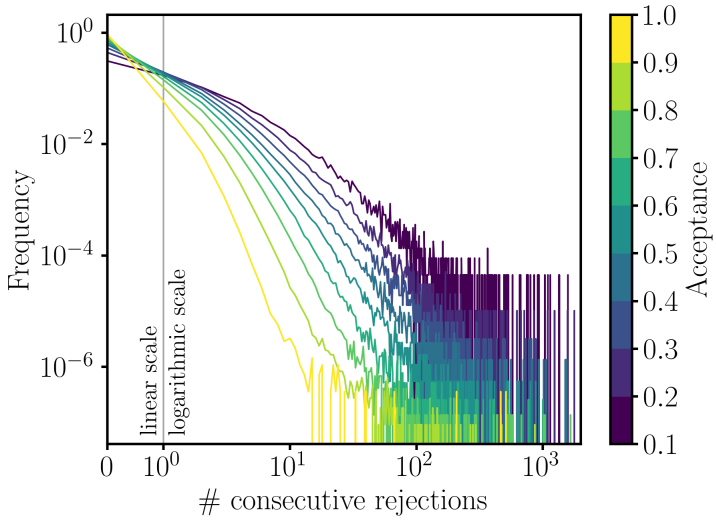

Figure 3.7 *Empirical distributions describing the length of periods of consecutive rejections in MH sampling involving models with different average acceptance rates. Each empirical distribution combines accept/reject statistics from models with various combinations of layers, trained against different target theories (in terms of lattice size and couplings). The scale on the x-axis is logarithmic other than between zero and one, zero consecutive rejections refers to two successive candidates being accepted.*

3.6.3 Finding efficient representations

An important factor in the scaling of the training costs is the number of trainable parameters, $|\theta|$. The game is to build an expressive enough flow with the smallest amount of redundancy. We want to find the most efficient representations of trivialising maps, where $|\theta|$ grows slowly as we increase the number of degrees of freedom in the target density.

In the case of the free theory, there was no advantage to using affine and rational quadratic spline transformations over the much simpler additive transformations. However, now that we turn our attention to interaction theories we can examine the role of the different transformations, at least empirically.

As previously alluded to, we are able to sample from the intermediate layers, which define probability densities in their own right, an example is given in Fig. 3.8 where we show histograms of the field variables and the two point correlation function from Eq. 3.41 on configurations outputted by intermediate layers of the flow model. To be a bit more precise, for a flow model defined with several

coupling layers

$$f \equiv g_I \circ g_{I-1} \circ \ldots \circ g_2 \circ g_1 \ , \tag{3.51}$$

we can generate a set of latent configurations $z \sim r(z)$ and then examine the partially transformed configurations, *i.e.* the configurations after the first coupling layer $g_1(z)$, the second coupling layer $g_2(g_1(z))$ etc. This gives us insight into the role played by each individual layer.

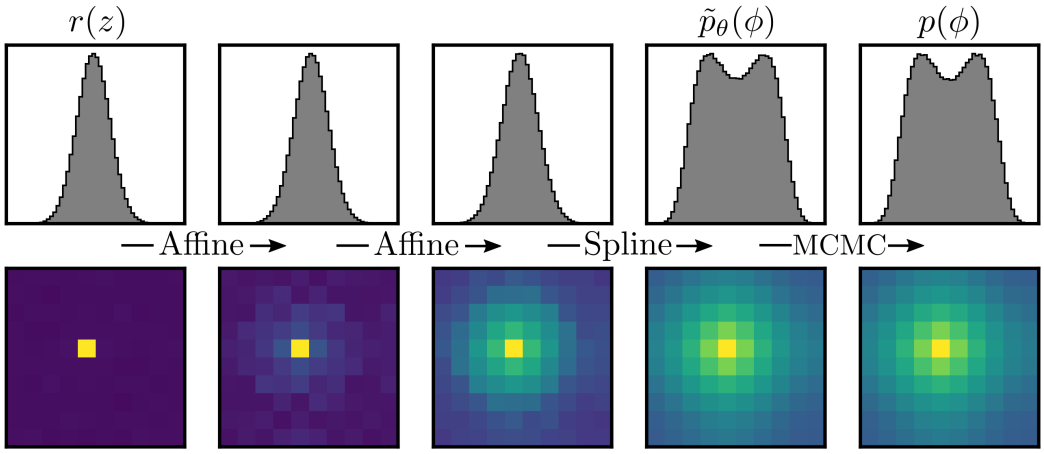

Figure 3.8 *Samples of field configurations drawn from each step in the process of generating a representative sample of ϕ^4 configurations with $\{L, \beta, \lambda\} = \{10, 0.601, 0.5\}$, using a flow model consisting of two affine coupling blocks, followed by a spline. The process starts with uncorrelated latent variables, $z \equiv v_1$, and we then sample from the model after each coupling block (layers $i = 3, 5, 7$), and finally take the output of the Metropolis-Hastings phase (keeping one in every $2\tau_{int}$ configurations). The top row contains histograms of the field variables and the bottom is the two point correlation function from Eq. 3.41. The emerging correlations are emphasised by the linear and logarithmic scaling of the colour map (a symmetric log scaling with a linear threshold of 0.1).*

In the regime where the distribution of the field variables is bimodal, with states having either net positive or net negative magnetisation, we find that the separation of the unimodal latent variable into the bimodal field variable typically happens in the last coupling block when the coupling layers are all affine. It appears as if the flow first of all switches on the correlations between variables and then shifts bulks of the two modes to $\pm\langle|\phi|\rangle$. Conversely, when we introduced a RQS coupling block into the flow, it always took on the role of transforming the unimodal distribution to a bimodal distribution, regardless of its position in the flow. This behaviour can be observed in Fig. 3.8, although here the RQS block is positioned at the end

of the flow.

Despite the RQS layer always taking on the same role in the transformation we show in Fig. 3.9, at least with the combinations we tested, that the flow generally performs best when the RQS is the final layer.

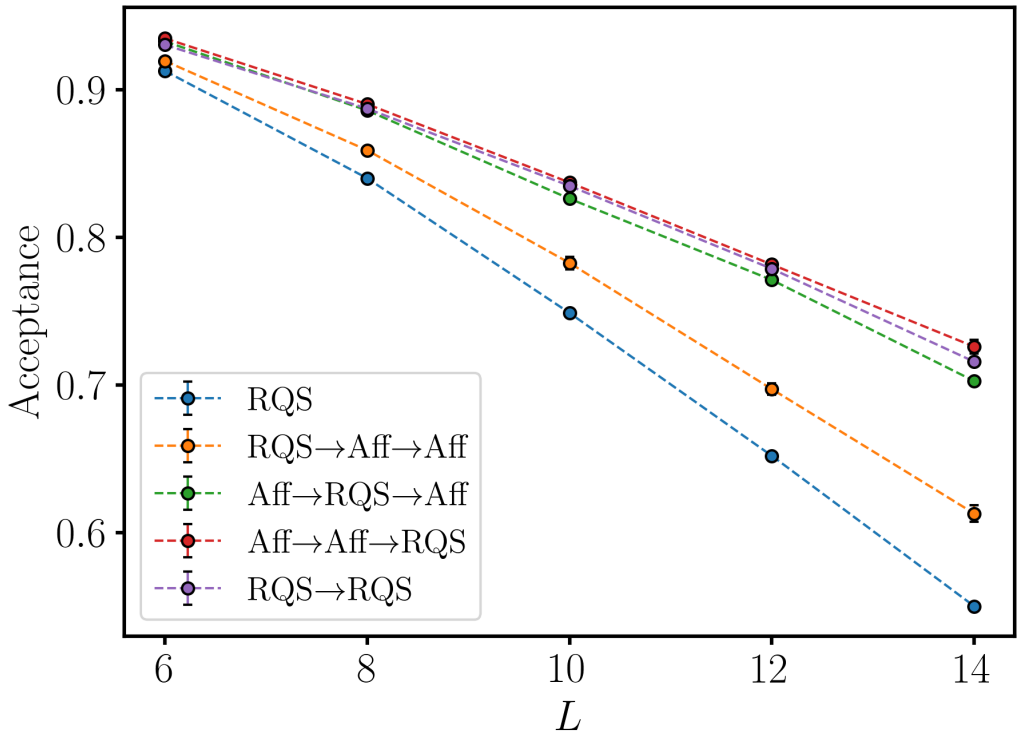

Figure 3.9 *Comparison of flow models with different arrangements of affine (Aff) and rational quadratic spline (RQS) coupling blocks. E.g. the blue model contains a single RQS block, i.e. two RQS coupling layers, whereas the orange model passes the latent variables through an RQS block, then two affine blocks. All neural networks contained a single hidden layer of size $H = |\Lambda|$. All models were trained for 16000 iterations with a batch size of 16000.*

Our understanding is that the additional flexibility of the RQS might help it perform the unimodal-bimodal split better than the affine layers, but the affine layers earlier in the flow help to build in the correct correlations between field variables.

This is further supported by comparing pure affine flows to affine-spline flows with approximately the same number of trainable parameters in Fig. 3.10. Here we took a seven-block affine flow and substituted six blocks of affine coupling layers for a single RQS block. Note, however, that this trick cannot be repeated. In Fig. 3.9

we show that adding a second spline block fails to improve the acceptance rate any more than adding another affine block. It appears that, with the unimodal-bimodal transformation taken care of in the final layer, the remainder of the trivializing map can be modelled more efficiently using the simpler transformations.

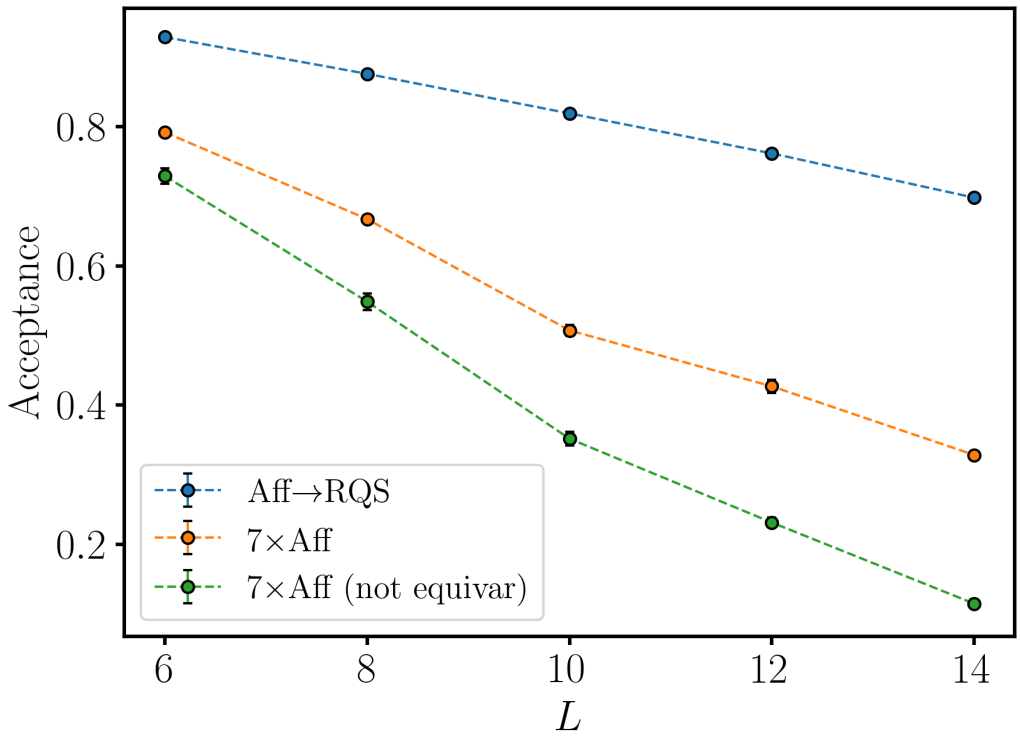

Figure 3.10 *Comparison between three groups of flow models using different coupling layers. One RQS block contains approximately the same number of parameters as six affine blocks. 'Equivar' refers to the $\mathbb{Z}_2$-equivariant affine layers described in Sec. 3.3.5. The green data points correspond to flows in which the affine layers were similar to those used in [1]. Models were trained in an identical fashion, for 16000 iterations with a batch size of 16000.*

It's clear in Fig. 3.10 that enforcing $\mathbb{Z}_2$-equivariance in the affine layers as described in Sec. 3.3.5 leads to higher acceptances which also scales slightly better with the lattice size. In Fig. 3.11, we see that the acceptance is only improved in the broken phase when $\mathbb{Z}_2$-equivariance is enforced (crudely, as per the prescription at the end of Sec. 3.3.5). Considering the potential risks with the crude implementation of enforcing $\mathbb{Z}_2$-equivariance in the RQS layers (potentially breaking ergodicity), we don't find a compelling reason for enforcing that symmetry in the RQS layers.

There are diminishing returns when adding more affine layers to the flow. The

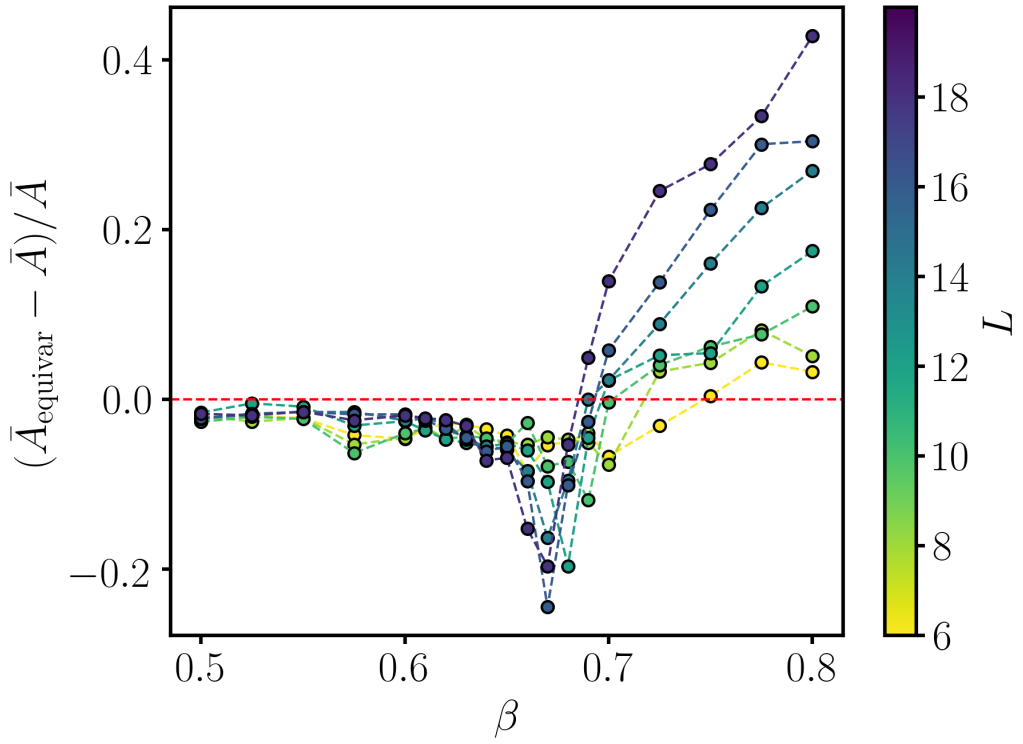

Figure 3.11 *Changes in the Metropolis-Hastings acceptance rate (denoted by $\bar{A}$) of spline-based models, due to enforcing $\mathbb{Z}_2$-equivariance in the spline layers, as described in Section 3.3.5.*

gains are most marked for larger lattices, but we see in Fig. 3.12 that just five affine coupling pairs is enough to reach maximum acceptance. Likewise, we found that adding more segments to the RQS had diminishing returns and that eight segments resulted in sufficiently flexibly RQS layers, without slowing down the training too much.

During numerous experiments, we found that using deep neural networks in the coupling layers was no better than shallow networks with just a single hidden layer. Some of the results of this experimentation is shown in Fig. 3.13, along with some experimentation with increasing the size of the hidden layer in the shallow networks. We found that the benefits of increasing the width of neural networks quickly diminished once the hidden layer contained more than $H = |\Lambda|$ elements (i.e. twice the size of the input layer, which is the passive partition only). Since a doubling of the neural network widths in spline layers increases the number of parameters by a factor of approximately $3K|\Lambda|$, incurring a considerable increase in training costs, it was almost never a better investment of resources than increasing the batch size or number of training iterations.

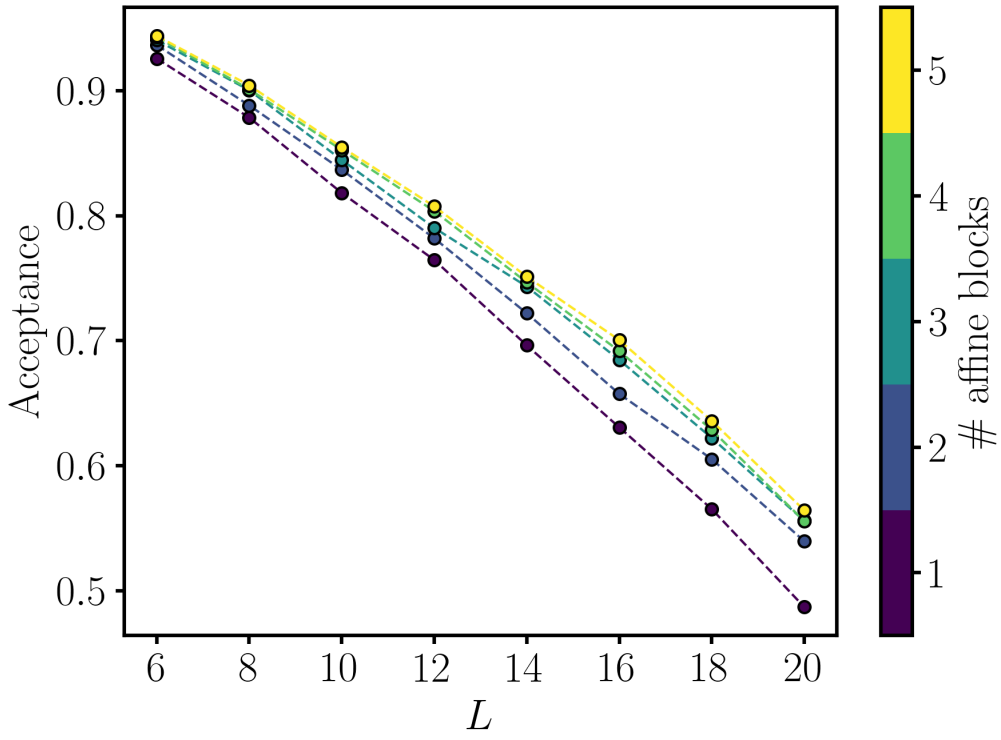

Figure 3.12 *Improvements in the Metropolis-Hastings acceptance rate due to adding more affine coupling blocks, before a rational quadratic spline block performs the final transformation.*

A stark feature of Fig. 3.13 demonstrates a key conclusion of this work. The acceptance rate of the models is strongly dependent on the cost of the training (the total number of states generated during the training $|\Phi_{\text{train}}|$) and is relatively oblivious to adding trainable parameters to the model.

3.6.4 Scaling of training costs

Neglecting algorithmic or hardware-related factors that contribute to scalability, the cost of training a model up to a given value in some performance metric (e.g. acceptance rate) can be measured in terms of the number of training iterations and the batch size. A single number which contains all of this information is the total number of configurations that the model has been exposed to during training, $|\Phi_{\text{train}}|$. This has the added advantage of being comparable to algorithms which suffer from critical slowing down, where the integrated autocorrelation time is proportional to the total number of configurations that must be generated to achieve a target error on expectation values. We provide some training times,

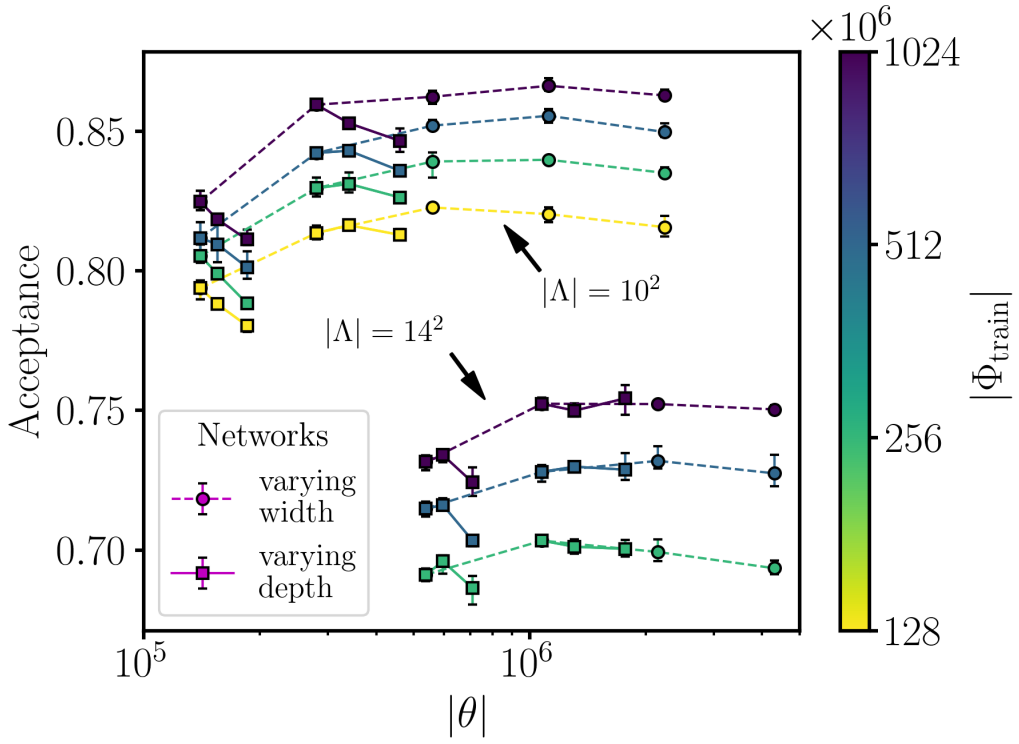

Figure 3.13 *Comparison of Metropolis-Hastings acceptance rates for flow models with a different number of trainable parameters, $|\theta|$, within their neural networks. The colour axis, $|\Phi_{train}|$, denotes the total number of configurations used in the optimisation (the batch size multiplied by the number of training iterations). The dotted lines connect models whose neural networks had a single hidden layer, of varying width, whereas following the dashed lines equates to increasing the number of hidden layers without changing the widths. Models comprised one affine block followed by one RQS block and were trained for 32000 training iterations with a range of batch sizes.*

measured in seconds in Appendix B.3, but stress the hardware-dependence of these numbers.

In Fig. 3.14 we plot the mean and standard deviation of the acceptance for models with the same training cost, $|\Phi_{\text{train}}|$. The model parameters for each lattice size are given in Table 3.1. In order to vary $|\Phi_{\text{train}}|$, we incremented both the minibatch size, between 250-32000, and the number of training iterations, between 500-64000 but not necessarily together. Single points in Fig. 3.14 are made up of a mixture of different batch sizes and training iterations. We find the small errorbars reassuring, indicating that the variable of importance is indeed $|\Phi_{\text{train}}|$. The acceptance rate does not appear to plateau even for the smallest lattice, which indicates that the

acceptance is not limited by the model saturating, but rather by the amount of training.

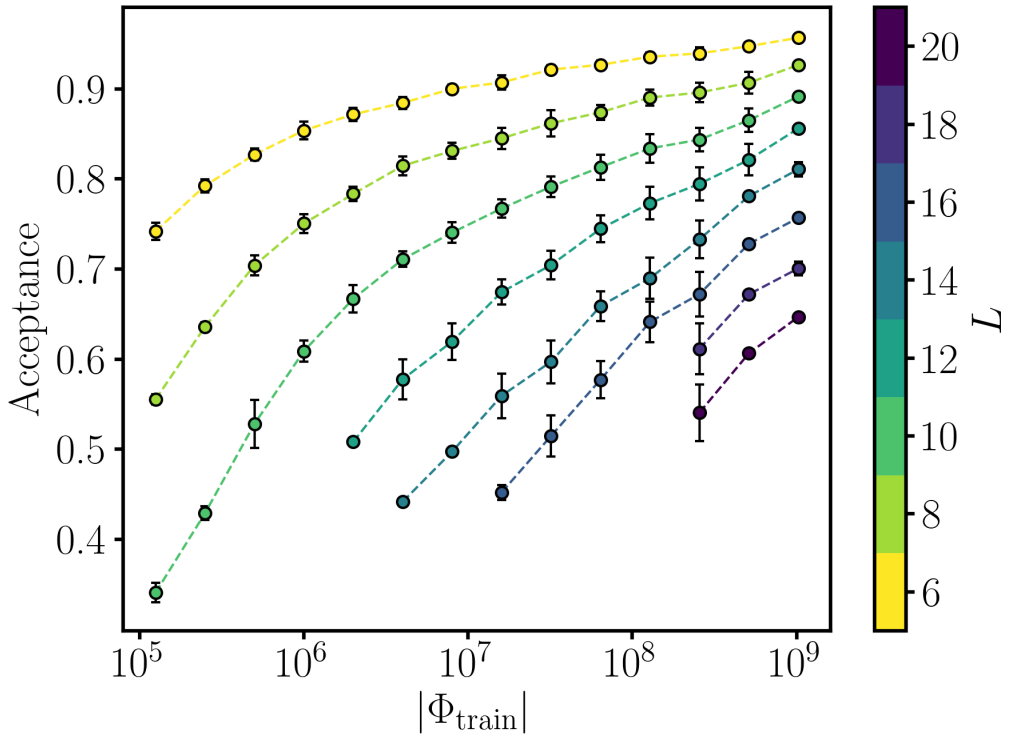

Figure 3.14 *Dependence of the average acceptance rate on the total number of configurations to which it has been exposed during the training phase, i.e. the product of the batch size and the number of training iterations. Error bars are standard deviations over a number of different models trained using different batch sizes and training lengths, as well as a variable number of affine layers.*

We make a first attempt to quantify the scaling of $|\Phi_{\text{train}}|$ in a way that can be contrasted with critical slowing down. In order to get a set of points which could be fit using a power law, we trained a series of models whilst varying the size of the lattices and the total number of states sampled from the model during the training $|\Phi_{\text{train}}|$. We then grouped models by the integrated autocorrelation time estimated from the magnetisation in bins of width $\Delta\tau_{\text{int},M} = 0.25$. For each group (and lattice size, where available) we choose the model with the lowest integrated autocorrelation time and then fit the exponent of $|\Phi_{\text{train}}|$ vs. lattice size for models of approximately constant integrated autocorrelation time. In Fig. 3.15 we show the results of these fits, and note that despite the drastically reduced training time compared to the original study [1], the amount of effort required to train these models is growing at an astonishing rate. While it is true that $|\Phi_{\text{train}}|$ is a

one-off overhead cost, unlike the number of configurations required in a traditional sampling simulation, scaling that goes with the 9th power of the correlation length makes it impossible to avoid the conclusion that this prescription remains far away indeed from a solution to critical slowing down.

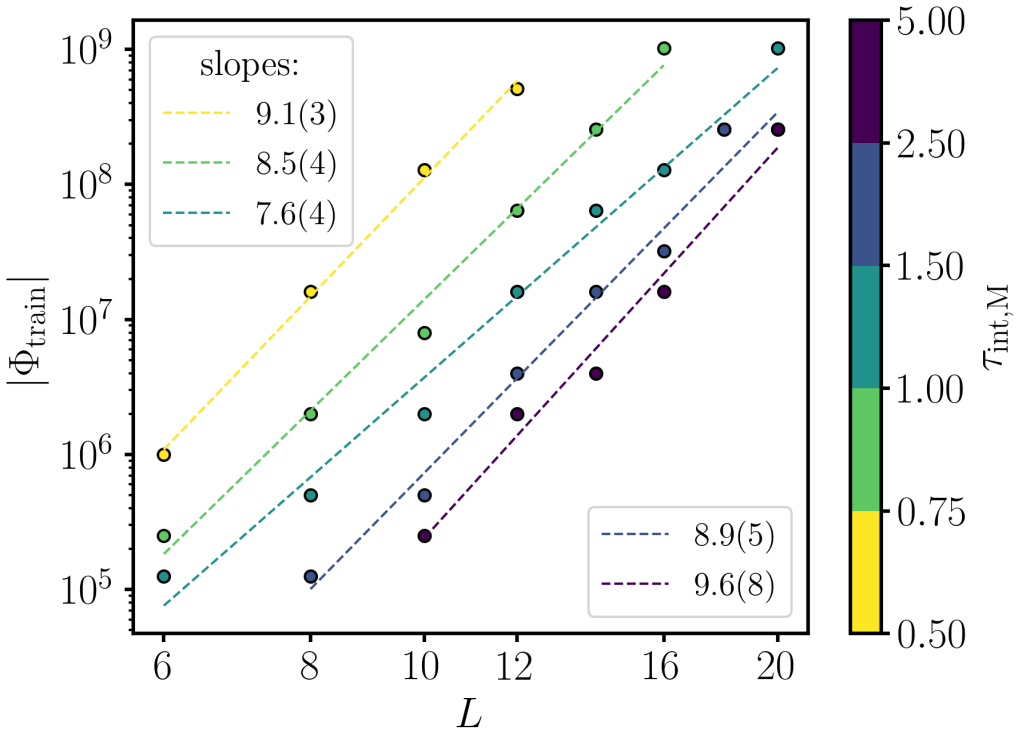

Figure 3.15 *Scaling of the training cost, measured by the total number of configurations used for optimisation, for models to reach a certain target sampling efficiency. Models were sorted into 'bins' by the integrated autocorrelation time measured during the sampling phase, and data points represent the best model from each bin. The values in the legend can in some sense be compared to the critical exponent $z_{\mathcal{O}}$ which determines the critical slowing down in traditional simulations.*

3.7 Discussion

Machine learning has, generally, followed a trend towards increasingly deep architectures, in spite of the fact that many of these architectures, including the fully-connected feed-forward neural networks defined used in this work, require only a single hidden layer to act as universal approximators. One motivation for preferring deep architectures is that they are less inclined to over-fit data [139] a potentially surprising feature which is explored further in [140, 141]. However, as already discussed Sec. 3.3.1, over-fitting is not a problem we will face since each training iteration exposes the model to a set of previously-unseen training inputs. There are also arguments that deep networks can represent highly non-linear functions more efficiently than shallow networks. Regardless, our experimentation appears to show that increasing the depth of the neural networks doesn't offer any benefit.

It is possible that this result is not merely a peculiar outcome of building the map out of coupling layers, whose structure is quite unusual; we draw attention to Ref. [86], in which the conclusion was that a single-layered restricted Boltzmann machine provides a more efficient representation of an Ising system near criticality than any of its deep generalisations.

Furthermore, given a fixed number of trainable parameters, we found that a shallow flow with a more flexible spline layer dramatically outperformed a deeper flow using affine layers, for the interacting theory at least.

Conventional training, via optimising Eq. 3.18 using a fixed training set, penalises the model for under-estimating the density at any point at which there is a training input. This tends to result in models that are smoothed approximations of $p(\phi)$, spanning the space of training inputs. If we were to use such a model as the basis of our sampling algorithm, we might expect to see a steady flux of configurations originating from regions of configuration space of low density in $p(\phi)$, which would typically be rejected by the Metropolis test. The situation for our models is quite different. Instead, the characteristic behaviour, which is known as *zero-forcing*, is for the optimisation to quickly purge the prior density from any region of configuration space in which $p(\phi)$ is very small, resulting in models that fit the modes of the target well but underestimate the low-density tails [142]. If we return the metropolis test in Eq. 3.9, we see that there is a factor in the acceptance probability associated with the model distribution divided by

the target distribution evaluated on the current state $\tilde{p}_\theta(\phi)/p(\phi)$. The point is that if a state gets accepted where the model distribution underestimates the target distribution, then this factor will suppress the probability of transitioning away. This explains the observed behaviour in the MH sampling step, where acceptance rate was typically high but featured rare periods of many consecutive rejections.

Fig. 3.16 shows how the training can be split into two phases. In the first phase the modes of the distribution are fitted rapidly, essentially by zero-forcing. In the second phase, the convergence is much slower, the process of expanding the modes to fit the tails is much more gradual. Recall that as the acceptance goes up during this second phase of the training, the distribution of consecutive rejections becomes less long-tailed, as demonstrated by Fig 3.7.

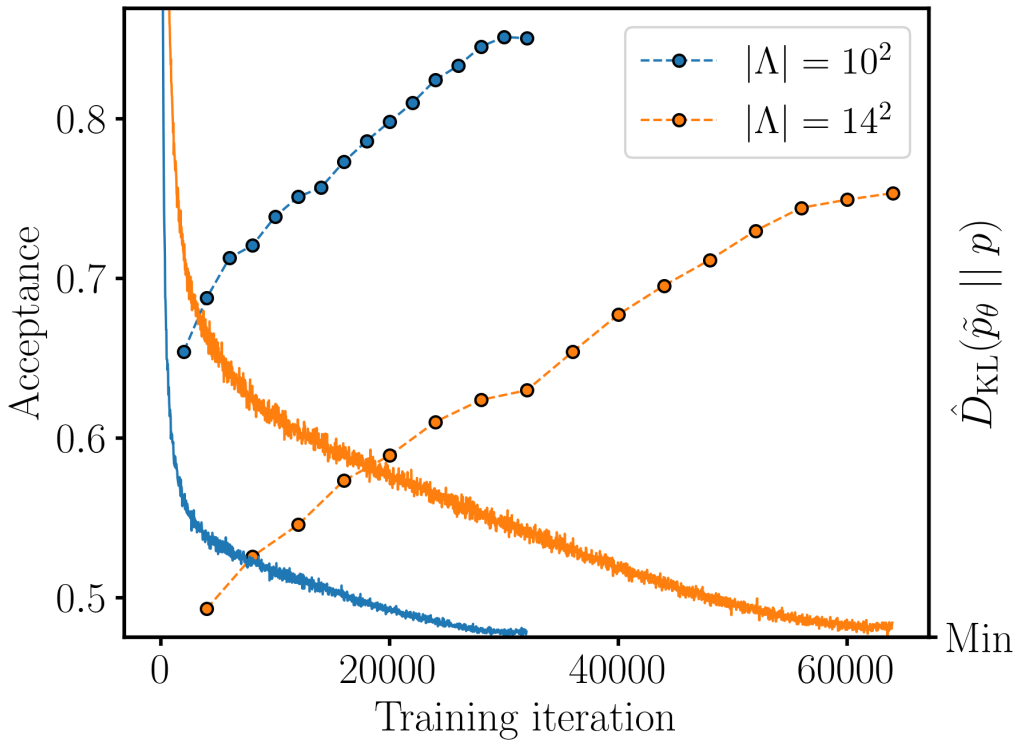

Figure 3.16 *Examples of typical training profiles for hybrid affine-spline flow models being trained against interacting theories with correlation length $\xi \approx L/4$. The objective function (right-axis) has been shifted for the purpose of fitting both profiles on one figure. The learning rate is being annealed down from η_0 to zero according to Equation* (3.47).

Something which we observed when training models in the broken phase was an

alarming feature of zero-forcing combined with a too high initial learning rate. In Fig. 3.17 we show an example of such a model, which has essentially broken $\mathbb{Z}_2$ symmetry by being trained too aggressively. Since the model rarely produces training samples in the tail of the distribution, where there should be a second mode, it's very difficult to recover a model which has fallen into this type of regime. The bottom plots show that this can be avoided by increasing the batch size and/or reducing the learning rate, in which case the initial phase of the training is kept under control.

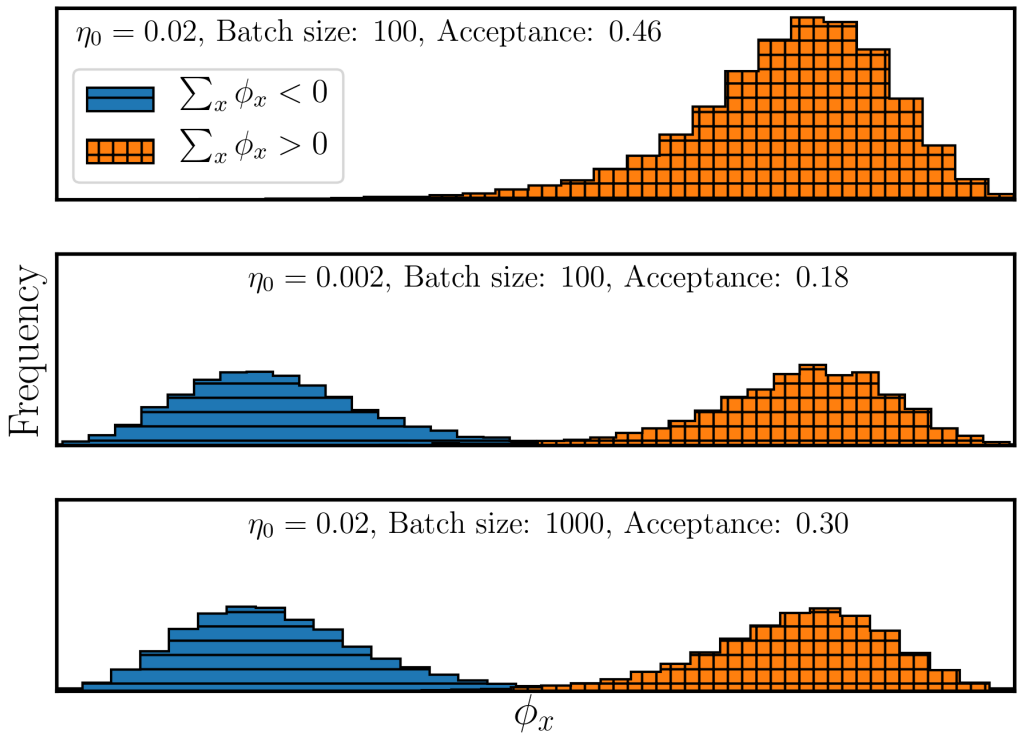

Figure 3.17 *Histograms of field variables taken from three samples of* 10^5 *configurations, generated by three different models. The colour labels the sign of the magnetisation. The* ϕ^4 *parameters are* $\{L, \beta, \lambda\} = \{6, 0.8, 0.5\}$. *The models had two blocks of affine layers for which* $\mathbb{Z}_2$*-equivariance was not enforced. In the top sub-figure, the large learning rate and small batch size result in the breaking of the* $\mathbb{Z}_2$ *symmetry during optimisation. This is easily avoided by using more sensible learning rates and batch sizes. However, note that the acceptance rate for the top model is larger.*

Note that the model in the top plot has a higher acceptance rate than both of the other models, despite it being incorrect. This might seem counter-intuitive but if the sampling was run for long enough, eventually a configuration from the collapsed mode would be generated, at which point the Markov chain would freeze

due to the enormously suppressed probability of transitioning away.

This should at least serve as a warning, to not just look at the MH acceptance as a performance indicator, but also check that the sample produced by the model doesn't violate any of the known symmetries of the system. For example, it's fairly clear from top plot of Fig. 3.17 that the $\mathbb{Z}_2$ symmetry has been broken, simply by looking at histograms of the field variables.

The key conclusion of our results is that the quality of trained models is limited by how extensively they have been optimised. For the ϕ^4 couplings given in Table 3.1 and our hybrid affine-spline models, Fig. 3.15 shows that these training costs scale in proportion to the number of degrees of freedom in the target density raised to a fairly high power — in the region of 7–10.

Larger lattice size unavoidably increases the number of trainable parameters, since we fixed the neural networks to have a hidden layer of size $H = |\Lambda|$, the size of models grows as $|\theta| \propto |\Lambda|^2 = L^4$. The results of Fig. 3.15 then state that, given a fixed target acceptance rate, the number of configurations $|\Phi_{\text{train}}|$ needed to train models scales with at least the square of the number of parameters requiring optimisation. The numbers here are less important than the fact that the poor scaling is not simply due to the increased number of parameters.

The point is that after the initial zero-forcing phase of the training, the probability of generating a configuration which produces a gradient, $\nabla_\theta \hat{D}_{\text{KL}}(\tilde{p}_\theta \,||\, p)$, in a region where the model distribution underestimates the target distribution is very small. During later stages in the training, once the model has learnt the modes of the target distribution, we are then reliant on sampling from the tail of the model distribution specifically in the sub-manifold of the target distribution where $\tilde{p}_\theta(\phi) \ll p_\theta(\phi)$. This is especially challenging for target distributions with singular covariance matrices [142]. For demonstrative purposes in Fig. 3.18 we show a schematic in two dimensions with a sample from a fictional model distribution at some point during the training and a sample from the corresponding target distribution (for which the two dimensions are clearly highly correlated). The model samples more frequently in regions where $\tilde{p}_\theta(\phi) \gg p_\theta(\phi)$ which has the zero-forcing effect of pushing the model distribution towards sampling the mode of the target more heavily. Conversely, the model rarely samples along the highly correlated direction of the target distribution, we believe that the process of learning this correlated direction is what takes an exponential amount of effort in the later stages of the training.

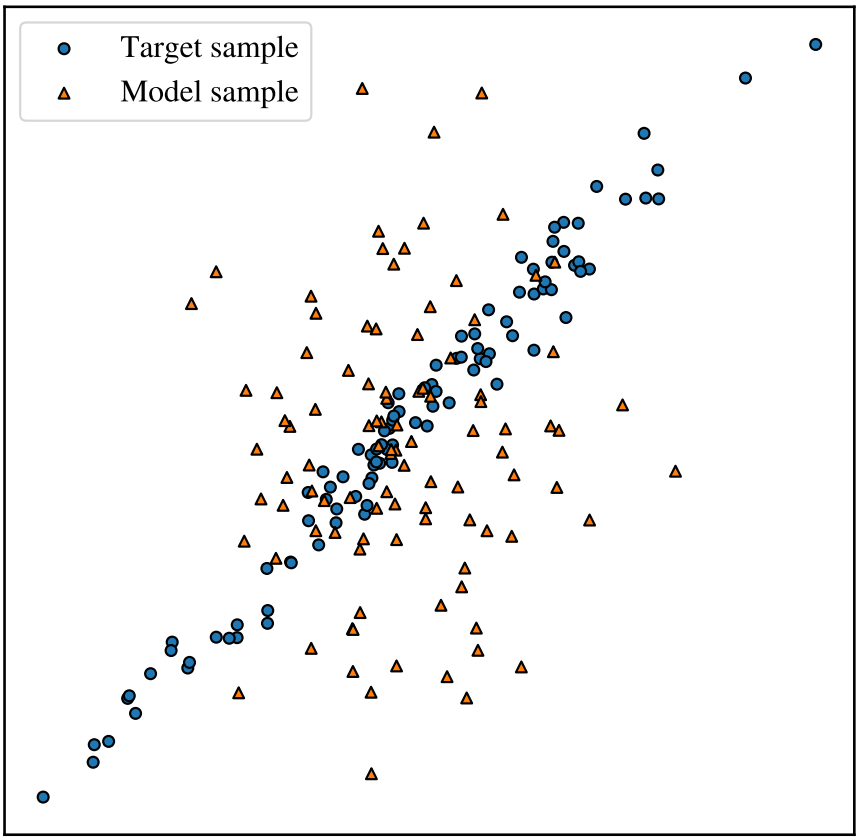

Figure 3.18 *Example of samples drawn from model during the training vs. a sample from a target distribution which has high correlations in the covariance matrix. The majority of samples from the model are in regions where $\tilde{p}_\theta(\phi) \gg p_\theta(\phi)$, and so have the zero-forcing effect. The direction for which the target distribution is highly correlated is rarely sampled by the model and so learning the tails of the target distribution in this direction has high training costs.*

3.8 Conclusions

We have verified the key results of Reference [1]: firstly, that there exist approximately trivializing maps in two-dimensional lattice ϕ^4 theory that are accessible *in practice* to normalizing flow models with tractable Jacobian determinant; and secondly, that using such models as generators of proposals for a Metropolis-Hastings simulation represents a complete transfer of the computational

costs normally associated with critical slowing down to the costs of training the flow model. As it often the case in Machine Learning, the efficiency of the procedure depends on the tuning of hyperparameters, which we have investigated in this work. We have shown that fairly modest modifications of the original prescription — inserting a more expressive transformation at the final layer of the flow and drastically reducing the size of the neural networks — lead to much more efficient representations of approximately trivializing maps for this system. The extent of the associated reduction in training costs is such that the systems studied here and in [1] are accessible with the sorts of computing resources typically found on personal computers.

However, our main finding is that the rate at which training costs scales as we move towards the continuum limit is extremely large, increasing far more quickly than the size of the models. Our work demonstrates a rather urgent need to understand and mitigate inefficiencies in the training algorithm itself when the target of optimisation is a lattice field theory in the critical regime, and this must be done in parallel with efforts towards building more sophisticated flow models.

Chapter 4

Bayesian Approach to inverse problems

4.1 Introduction

Inverse problems fall under the paradigm of supervised learning and are the typical example of inference where a model is sought starting from a finite-dimensional set of experimental observations. These problems are notoriously difficult, and often require trying to guess a continuous function, *i.e.* an element of an infinite dimensional space, from a finite amount of data. As emphasised by Hadamard a long time ago, it is easy to end up in a situation where we deal with ill-posed problems, in the sense that the solution may not exist, may not be unique, or may be unstable under small variations of the input data. The determination of parton distributions from experimental data, or the reconstruction of spectral densities from lattice QCD simulations, are just two examples where these problems arise in particle physics. In all these cases, finding a robust solution becomes very challenging, if not impossible, making these questions all the more urgent, especially at a time when precision studies are the ultimate challenge in order to highlight divergences from the Standard Model.

A Bayesian approach provides an apter tool for addressing inverse problems. Starting from a prior distribution for the model, which encodes our theoretical knowledge or prejudice, basic statistical tools allow us to determine the posterior distribution of the solution, *after taking into account a set of experimental*

observations. The prior and posterior probabilities encode our knowledge about the model before and after incorporating the experimental results. There are multiple advantages to this formulation: the inverse problem is well defined, the prior distribution ensures that all the assumptions are explicit, and it allows to regulate the likelihood distribution.

Note also that probability measures can be defined in infinite-dimensional spaces, In cases where we are looking to determine a continuous function, the Bayesian approach allows, at least in principle, to address the problem directly in terms of the posterior probability measure of the model function. It is often convenient for practical reasons to define a parametrization of the model function in terms of a finite, albeit large, number of parameters and reduce the problem to a finite dimensional one. The price to pay for this simplification is the introduction of some bias, which needs to be understood and possibly quantified. An expressive parametrization clearly helps in this case. The Bayesian approach is also well suited to include potentially inconsistent data in a systematic way, but we will only briefly touch upon this.

A Bayesian approach to inverse problems has been actively developed by mathematicians for a long time, and this development has accelerated quickly in the last decade. In this Chapter we aim at summarising the existing framework and adapting it in the context of analysing the fits of parton distribution functions obtained by the NNPDF collaboration. We will review the formalism in Sec. 4.2, trying to define the notation that will be used in the rest of the Chapter. We will report some examples already known, for illustrative purposes. We then try to connect the Bayesian approach with the NNPDF fits based on a Monte Carlo methodology, where the distribution of the PDFs is encoded in a set of fits to artificially generated data, called replicas. We can anticipate here that, under the hypotheses that the data are Gaussian and the model is linear, the NNPDF procedure would characterise completely the posterior probability density. When the model is non-linear two modifications need to be taken into account. First of all the analytical calculation that we present in Sec. 4.2.1 is no longer possible, and one needs to rely on the fact that a linearization of the model yields an accurate prediction. Whilst we believe that the linear models can provide a good approximation, the systematic errors introduced by this approximation are not easy to quantify. There is also a more subtle effect that needs to be taken into account. When working with linear models, the minimization procedure is know to have a unique minimum, which can be computed exactly. Non-linear models

can be plagued by multiple minima, and more importantly by inefficiencies of the algorithm in finding them. While it is important to be aware of the limitations of the analytical calculation, it is also very useful to have an explicit result as a template to guide the analysis of our numerical investigations.

In the NNPDF fitting framework, the posterior probability of the parton distribution functions, is encoded in an ensemble of fits to replicas of the data, where the data replicas have been generated in order to induce the fluctuations described by the experimental central values and uncertainties. This bootstrap procedure propagates the data fluctuations in the space of fitted PDFs. A successful fit should yield robust confidence intervals for observables, in particular for those that are more relevant for phenomenology.

The idea of *closure tests* is to test this procedure in a fit to artificial data that have been generated from a known set of input PDFs. In this case the underlying law is known and we can check how the posterior distribution compares to the underlying law. This is the basis of a closure test, which has already been used to test to validity of previous iterations of the NNPDF methodology. Here we aim to refine some of the pre-existing closure test estimators and with the help of fast fitting methodology perform a more extensive study of how faithful our uncertainties are. For this purpose we introduce new estimators that allow us to quantify the quality of our fits. These estimators are defined in the space of data, and we can use the Bayesian formalism in order to compute analytically their probability distributions.

We emphasise the stark contrast between an approach to inverse problems which aims at sampling from the posterior distribution in model space, and a more classical regression exercise. The link between the two methods will be explored, but when the goal shifts from finding the single best fit to data to a distribution in model space then our interpretation of various statistical estimators also changes. Furthermore, once we think about distributions in model space, we start to move closer towards the generative models discussed in the previous Chapters, rather than the typical tools which fall under the paradigm of supervised learning. In the final Chapter of this Thesis, the connection between the work in this Chapter and Chapter 3 will be explored in a bit more detail, which could motivate future work which aimed to apply the normalizing flows to inverse Bayesian problems.

4.2 Inverse Problems

The problem of determining PDFs from a set of experimental data falls under the general category of *inverse problems*, *i.e.* the problem of finding the input to a given model knowing a set of observations, which are often finite and noisy. In this Section we are going to review the Bayesian formulation of inverse problems. It is impossible to do justice to this vast subject here. Instead we try to emphasise the aspects that are relevant for quantifying uncertainties on PDF determinations.

4.2.1 Statement of the problem

The space of inputs is denoted by X, while R denotes the space of responses. The model is specified by a *forward map*

$$\begin{aligned} G: \; & X \to R \\ & u \mapsto r = G(u)\,, \end{aligned} \tag{4.1}$$

which associates a response $r \in R$ to the input $u \in X$, where we assume that X and R are Banach spaces. [1] As an example we can think of u as being a Parton Distribution Function, *i.e.* a function defined on the interval $[0,1]$, and r a DIS structure function, which is related to the PDF by a factorization formula involving perturbative coefficient functions [143]:

$$r(x,Q^2) = \int_x^1 \frac{dz}{z}\, C(z,Q^2) u(x/z,Q^2)\,. \tag{4.2}$$

Note that in this example the forward map maps one real function into another real function. Experiments will not have access to the full function r but only to a subset of $N_{\rm data}$ observations. In order to have a formal mathematical expression that takes into account the fact that we have a finite number of measurements, we introduce an *observation operator*

$$\begin{aligned} O: \; & R \to Y \\ & r \mapsto y\,, \end{aligned} \tag{4.3}$$

[1] Banach spaces are complete normed vector spaces. We do not need to get into a more detailed discussion here, but it is important to note that working in Banach spaces allows us to generalise the results to infinite-dimensional spaces of functions.

where $y \in Y$ is a vector in a finite-dimensional space Y of experimental results, *e.g.* the value of the structure function for some values of the kinematic variables x and Q^2. In general we will assume that $y \in \mathbb{R}^{N_{\text{data}}}$, *i.e.* we have a finite number N_{data} of real experimental values. The quantity of interest is the composed operator

$$\begin{aligned} \mathcal{G} :\ & X \to \mathbb{R}^{N_{\text{data}}} \\ & \mathcal{G} = O \circ G \,, \end{aligned} \tag{4.4}$$

which maps the input u to the set of data. Taking into account the fact that experimental data are subject to noise, we can write

$$y = \mathcal{G}(u) + \eta \,, \tag{4.5}$$

where η is a random variable defined over $\mathbb{R}^{N_{\text{data}}}$ with probability density $\rho(\eta)$. We will refer to η as the *observational noise*. In this setting, the inverse problem becomes finding u given y.

In solving this problem we are going to adopt a Bayesian point of view, which can be summarised as follows: our prior knowledge about u is encoded in a prior probability measure μ_M^0, where the suffix M indicates that the measure is defined in the space of models, and the suffix 0 refers to the fact that this is a prior distribution. We use Bayes' theorem to compute the posterior probability measure of u given the data y, which we denote as $\mu_M^{\mathcal{G}}$. When possible, we denote the probability densities associated to μ_M^0 and $\mu_M^{\mathcal{G}}$, by π_M^0 and $\pi_M^{\mathcal{G}}$ respectively. Then, using Eq. (4.5), we can write the data likelihood, *i.e.* the probability density of y given u,

$$\pi(y|u) = \rho(y - \mathcal{G}(u)) \,, \tag{4.6}$$

and Bayes' theorem yields

$$\pi_M^{\mathcal{G}}(u) = \pi(u|y) \propto \rho(y - \mathcal{G}(u)) \pi_M^0(u) \,. \tag{4.7}$$

Example Even though the concepts that we have introduced so far should sound familiar, it is worthwhile to spend a few paragraphs to clarify some ideas and present an explicit example, where all the probability densities are carefully defined. This is best exemplified by considering the case where both the observational noise and the model prior are Gaussian. We assume that we are given a set of central

values $y_0 \in \mathbb{R}^{N_{\text{data}}}$ and their covariance matrix C_D. Then the *prior* probability density of the observable y is

$$\pi_D^0(y|y_0, C_D) \propto \exp\left(-\frac{1}{2}\,|y - y_0|^2_{C_D}\right) , \tag{4.8}$$

where, similarly to the convention used above, the suffix D emphasises the fact that this is a probability density in data space, and the notation explicitly reminds us that this is the probability density given the central values y_0 (and the covariance matrix). Similarly we can choose a Gaussian distribution for the prior distribution of the input model, characterized by a central value u_0 and a covariance C_M:

$$\pi_M^0(u|u_0, C_M) \propto \exp\left(-\frac{1}{2}\,|u - u_0|^2_{C_M}\right) . \tag{4.9}$$

Following the convention above, we use a suffix M here to remind the reader that we are looking at a probability density in the space of models. Note that in the expressions above we used the norms in X and $\mathbb{R}^{N_{\text{data}}}$ respectively, and introduced the short-hand notation

$$|a|^2_M = \left|M^{-1/2}a\right|^2 , \tag{4.10}$$

where a denotes a generic element of X, R or $\mathbb{R}^{N_{\text{data}}}$. For the case where $a \in \mathbb{R}^{N_{\text{data}}}$, we use the Euclidean norm and

$$|a|^2_M = \sum_{i,j} a_i M_{ij} a_j \,, \tag{4.11}$$

where the indices i, j run from 1 to N_{data}. Up to this point data and models are completely independent, and the joint distribution is simply the product of π_D^0 and π_M^0.

The forward map induces a correlation between the input model and the observables, so we introduce a probability density θ that describes these correlations due to the physical theory,

$$\theta(y, u|\mathcal{G}) = \delta\left(y - \mathcal{G}(u)\right) , \tag{4.12}$$

where the Dirac delta corresponds to the case where there are no theoretical uncertainties. Theoretical uncertainties can be introduced by broadening the distribution of y away from the exact prediction of the forward map, *e.g.* using a

Gaussian with covariance C_T,

$$\theta(y, u|\mathcal{G}) = \exp\left(-\frac{1}{2}\left|y - \mathcal{G}(u)\right|^2_{C_T}\right) . \tag{4.13}$$

Note however that there are no rigorous arguments favouring the fact that theoretical errors are normally distributed. Taking the correlation $\theta(y, u|\mathcal{G})$ into account, the joint distribution of y and u is

$$\pi^{\mathcal{G}}(y, u|y_0, C_D, u_0, C_M) \propto \pi^0_D(y|y_0, C_D)\pi^0_M(u|u_0, C_M)\theta(y, u|\mathcal{G}) . \tag{4.14}$$

We can now marginalize with respect to y, neglecting theory errors,

$$\begin{aligned}
\pi^{\mathcal{G}}_M(u|y_0, C_D, u_0, C_M) &\propto \int dy\, \pi^0_D(y|y_0, C_D)\pi^0_M(u|u_0, C_M)\theta(y, u|\mathcal{G}) && (4.15)\\
&\propto \pi^0_M(u|u_0, C_M) \int dy\, \pi^0_D(y|y_0, C_D)\delta\left(y - \mathcal{G}(u)\right) && (4.16)\\
&\propto \pi^0_M(u|u_0, C_M)\pi^0_D(\mathcal{G}(u)|y_0, C_D) . && (4.17)
\end{aligned}$$

We see that we have recovered Eq. 4.7. The log-likelihood in the Gaussian case is simply the χ^2 of the data, y_0, to the theory prediction, $\mathcal{G}(u)$:

$$-\log \pi^0_D(\mathcal{G}(u)|y_0) = \frac{1}{2}\left|\mathcal{G}(u) - y_0\right|^2_{C_D} . \tag{4.18}$$

In the notation of Eq. 4.7

$$\pi^0_D(\mathcal{G}(u)|y_0) = \rho\left(\mathcal{G}(u) - y_0\right) , \tag{4.19}$$

where in this case

$$\rho(\eta) \propto \exp\left(-\frac{1}{2}\left|\eta\right|^2_{C_D}\right) . \tag{4.20}$$

The probability density $\pi^{\mathcal{G}}_M(u|y_0, C_D, u_0, C_M)$ was called $\pi^{\mathcal{G}}_M(u)$ in Eq. 4.7, where the suffix $\mathcal{G}$ is a short-hand to denote the posterior probability in model space, taking into account all the conditional variables. Hence, for the Gaussian case, the result from Bayes' theorem reduces to

$$\begin{aligned}
\pi^{\mathcal{G}}_M(u) &\propto \exp\left[-\frac{1}{2}\left|y_0 - \mathcal{G}(u)\right|^2_{C_D} - \frac{1}{2}\left|u - u_0\right|^2_{C_M}\right] && (4.21)\\
&= \exp\left[-S(u)\right] . && (4.22)
\end{aligned}$$

Note that in the argument of the likelihood function we have the central values of the data points y_0. Eq. (4.21) is the Bayesian answer to the inverse problem, our knowledge of the model u is encoded in the probability measure $\mu_M^{\mathcal{G}}$, which is fully specified by the density $\pi_M^{\mathcal{G}}$. There are several ways to characterise a probability distribution. The NNPDF approach is focused on the determination of the *Maximum A Posteriori (MAP)* estimator, *i.e.* the element $u_* \in X$ that maximises $\pi_M^{\mathcal{G}}(u)$:

$$u_* = \arg\min_{u \in X} \left(-\frac{1}{2} |y_0 - \mathcal{G}(u)|^2_{C_D} - \frac{1}{2} |u - u_0|^2_{C_M} \right) . \tag{4.23}$$

For every instance of the data y_0, the MAP estimator is computed by minimising a regulated χ^2, we will refer to this procedure as the *classical fit* of experimental data to a model. Note that in the Bayesian approach, the regulator appears naturally after having specified carefully all the assumptions that enter in the prior. In this specific example the regulator arises from the Gaussian prior for the model input u, which is normally distributed around a solution u_0. The MAP estimator provides the explicit connection between the Bayesian approach and the classical fit.

4.2.2 Comparison with classical fitting

There are a number of results that make the connection between the two approaches more quantitative, and therefore more transparent. We are going to summarise these results here without proofs, referring the reader to the mathematical literature for the missing details. Working in the finite-dimensional case, we assume

$$u \in \mathbb{R}^{N_{\text{model}}} ,$$
$$y \in \mathbb{R}^{N_{\text{data}}} ,$$

and we are going to consider in detail two examples from Ref. [144], which illustrate the role of the priors in the Bayesian setting. It is particularly useful to distinguish the case of an underdetermined system from the case of an overdetermined one.

Underdetermined system The first case that we are going to analyse is the case of a linear system that is underdetermined by the data. The linear model is

completely specified by a vector of coefficients $g \in \mathbb{R}^{N_{\text{model}}}$,

$$\mathcal{G}(u) = \left(g^T u\right) . \tag{4.24}$$

Assuming that we have only one datapoint, *i.e.* $N_{\text{data}} = 1$,

$$y = (g^T u) + \eta \,, \tag{4.25}$$

where $\eta \sim \mathcal{N}(0, \gamma^2)$ is one Gaussian number, whose probability density is centred at 0 and has variance γ^2. For simplicity we are going to assume that the prior on u is also a multi-dimensional Gaussian, centred at 0 with covariance matrix C_M. In this case the posterior distribution can be written as

$$\pi_M^{\mathcal{G}}(u) \propto \exp\left[-\frac{1}{2\gamma^2}\left|y - (g^T u)\right|^2 - \frac{1}{2}\left|u\right|^2_{C_M}\right] , \tag{4.26}$$

which is still a Gaussian distribution for u. The mean and covariance are respectively

$$m = \frac{(C_M g) y}{\gamma^2 + (g^T C_M g)} \,, \tag{4.27}$$

$$\Sigma = C_M - \frac{(C_M g)(C_M g)^T}{\gamma^2 + (g^T C_M g)} \,. \tag{4.28}$$

Because the argument of the exponential is a quadratic form, the mean of the distribution coincides with the MAP estimator. It is instructive to look at these quantities in the limit of infinitely precise data, *i.e.* in the limit $\gamma \to 0$:

$$m_\star = \lim_{\gamma \to 0} m = \frac{(C_M g) y}{(g^T C_M g)} \,, \tag{4.29}$$

$$\Sigma_\star = \lim_{\gamma \to 0} \Sigma = C_M - \frac{(C_M g)(C_M g)^T}{(g^T C_M g)} \,. \tag{4.30}$$

These values satisfy

$$(g^T m_\star) = y \,, \tag{4.31}$$

$$(\Sigma_\star g) = 0 \,, \tag{4.32}$$

which shows that the mean of the distribution is such that the data point is exactly reproduced by the model, and that the uncertainty in the direction defined by g vanishes. It should be noted that the uncertainty in directions perpendicular to g does not vanish and is determined by a combination of the prior and the model,

viz. C_M and g in our example. This is a particular example of a more general feature: for underdetermined systems the information from the prior still shapes the probability distribution of the solution even in the small noise limit.

Overdetermined system We are now going to consider an example of an overdetermined system and discuss again the case of small observational noise. We consider $N_{\text{data}} \geq 2$ and $n = 1$, with a linear forward map such that

$$y = gu + \eta \,, \tag{4.33}$$

where η is an N_{data}-dimensional Gaussian variable with a diagonal covariance $\gamma^2 I$, where I denotes the identity matrix. For simplicity we are going to assume a Gaussian prior with unit variance for u, which yields for the posterior distribution:

$$\pi_M^{\mathcal{G}}(u) \propto \exp\left(-\frac{1}{2\gamma^2}\left|y - g(u + \beta u^3)\right|^2 - \frac{1}{2}u^2\right) . \tag{4.34}$$

If $\beta = 0$ the posterior is Gaussian and we can easily compute its mean and variance:

$$m = \frac{(g^T y)}{\gamma^2 + |g|^2} \,, \tag{4.35}$$

$$\sigma^2 = \frac{\gamma^2}{\gamma^2 + |g|^2} \,. \tag{4.36}$$

In this case, in the limit of vanishing observational noise, we obtain

$$m_\star = \frac{(g^T y)}{|g|^2} \,, \tag{4.37}$$

$$\sigma_\star^2 = 0 \,. \tag{4.38}$$

The mean is given by the weighted average of the datapoints, which is also the solution of the χ^2 minimization

$$m_\star = \arg\min_{u \in \mathbb{R}} |y - gu|^2 \,. \tag{4.39}$$

Note that in this case the variance $\sigma_\star$ vanishes independently of the prior. In the limit of small noise, the distribution tends to a Dirac delta around the value of the MAP estimator.

4.2.3 Linear Problems

Linear problems in finite-dimensional spaces are characterized by a simple forward law,

$$y = \mathcal{G}u\,, \tag{4.40}$$

where $\mathcal{G}$ is a matrix. In this framework one can readily derive analytical solutions that are useful to understand the main features of the Bayesian approach. Assuming that the priors are Gaussian again, the cost function $S(u)$ is a quadratic function of u,

$$\begin{aligned} S(u) &= (\mathcal{G}u - y_0)^T C_D^{-1} (\mathcal{G}u - y_0) + (u - u_0)^T C_M^{-1} (u - u_0) && (4.41)\\ &= (u - \tilde{u})\, \tilde{C}_M^{-1} (u - \tilde{u}) + \text{const}\,, && (4.42) \end{aligned}$$

where

$$\begin{aligned} \tilde{C}_M^{-1} &= \left(\mathcal{G}^T C_D^{-1} \mathcal{G} + C_M^{-1}\right)\,, && (4.43)\\ \tilde{u} &= \tilde{C}_M \left(\mathcal{G}^T C_D^{-1} y_0 + C_M^{-1} u_0\right)\,. && (4.44) \end{aligned}$$

The case where we have no prior information on the model is recovered by taking the limit $C_M^{-1} \to 0$, which yields

$$\begin{aligned} \tilde{C}_M^{-1} &= \left(\mathcal{G}^T C_D^{-1} \mathcal{G}\right)\,, && (4.45)\\ \tilde{u} &= \tilde{C}_M \left(\mathcal{G}^T C_D^{-1} y_0\right)\,. && (4.46) \end{aligned}$$

The action of C_D^{-1} on the vector of observed data y_0 is best visualised using a spectral decomposition

$$C_D^{-1} = \sum_n \frac{1}{\sigma_n^2} P_n\,, \tag{4.47}$$

where P_n denotes the projector on the n-th eigenvector of C_D, and σ_n^2 is the corresponding eigenvalue. The action of C_D^{-1} is to perform a weighted average of the components of y_0 in the directions of the eigenvectors of C_D.

An explicit expression for the posterior distribution of data can be obtained from

the joint distribution by marginalising over the model input u:

$$\pi_D^{\mathcal{G}}(y|y_0, C_D, u_0, C_M) = \int du\, \pi^{\mathcal{G}}(y, u|y_0, C_D, u_0, C_M) \tag{4.48}$$

$$\propto \exp\left(-\frac{1}{2}\left(y-\tilde{y}\right)^T \tilde{C}_D^{-1}\left(y-\tilde{y}\right)\right), \tag{4.49}$$

where

$$\tilde{y} = \mathcal{G}\tilde{u}\,, \tag{4.50}$$

$$\tilde{C}_D = \mathcal{G}\tilde{C}_M\mathcal{G}^T\,. \tag{4.51}$$

Posterior distribution of unseen data In real-life cases we are also interested in the posterior distribution of a set of data that have not been included in the fit. In the Bayesian framework that we have developed this can be modeled by having two sets of data y and y', for which we have a prior distribution

$$\pi_D^0\left(y, y'|y_0, C_D, y_0', C_D'\right) = \pi_D^0\left(y'|y_0', C_D'\right)\pi_D^0\left(y|y_0, C_D\right) \tag{4.52}$$

$$\propto \exp\left[-\frac{1}{2}\left(y'-y_0'\right)^T\left(C_D'\right)^{-1}\left(y'-y_0'\right)\right]\exp\left[-\frac{1}{2}\left(y-y_0\right)^T\left(C_D\right)^{-1}\left(y-y_0\right)\right]. \tag{4.53}$$

Following the derivation above, we can write the joint distribution for the data and the model

$$\pi^{\mathcal{G}}(y, y', u) \propto \pi_D^0(y, y'|y_0, C_D, y_0', C_D')\pi_M^0(u)\delta\left(y-\mathcal{G}u\right)\delta\left(y'-\mathcal{G}'u\right). \tag{4.54}$$

Note that because both sets of data are derived from the same model u, the joint distribution above introduces a correlation between the data sets.

We can now marginalise with respect to the dataset y,

$$\pi(y', u) \propto \exp\left[-\frac{1}{2}\left(y'-y_0'\right)^T\left(C_D'\right)^{-1}\left(y'-y_0'\right)\right]\exp\left[-\frac{1}{2}\left(u-\tilde{u}\right)^T\left(\tilde{C}_M\right)^{-1}\left(u-\tilde{u}\right)\right] \times \delta\left(y'-\mathcal{G}'u\right). \tag{4.55}$$

where $\tilde{C}_M$ and $\tilde{u}$ are given respectively in Eqs. 4.43 and 4.44. By marginalising again, this time with respect to the model, we derive the posterior distribution of

the unseen data,

$$\pi_D^y(y') \propto \exp\left[\frac{1}{2}\left(y'-\tilde{y}'\right)^T\left(\tilde{C}'_D\right)^{-1}\left(y'-\tilde{y}'\right)\right], \tag{4.56}$$

where

$$\tilde{C}'_D = \mathcal{G}'\tilde{C}'_M\mathcal{G}'^T \tag{4.57}$$

$$\tilde{y}' = \mathcal{G}'\tilde{u}', \tag{4.58}$$

and

$$\tilde{C}'^{-1}_M = \mathcal{G}'^T C'^{-1}_D \mathcal{G}' + \tilde{C}^{-1}_M \tag{4.59}$$

$$\tilde{u}' = \tilde{C}'_M\left(\mathcal{G}'^T C'^{-1}_D y'_0 + \tilde{C}^{-1}_M \tilde{u}\right) \tag{4.60}$$

$$\tilde{C}^{-1}_M = \mathcal{G}^T C^{-1}_D \mathcal{G} + C^{-1}_M \tag{4.61}$$

$$\tilde{u} = \tilde{C}_M\left(\mathcal{G}^T C^{-1}_D y_0 + C^{-1}_M u_0\right). \tag{4.62}$$

A comment on non-linear models The linear models that we have discussed so far may look over-simplified at first sight. In practice, it turns out that non-linear models can often be linearised around the central value of the prior distribution,

$$\mathcal{G}(u) = \mathcal{G}(u_0) + G\left(u-u_0\right) + \dots, \tag{4.63}$$

where

$$G^i_\alpha = \left.\frac{\partial \mathcal{G}^i}{\partial u_\alpha}\right|_{u_0}, \tag{4.64}$$

and we have neglected higher-order terms in the expansion of $\mathcal{G}(u)$.

If these terms are not negligible, another option is to find the MAP estimator, and then expand the the forward map around it, which yields equations very similar to the previous ones, with u_0 replaced by u_*. If the posterior distribution of u is sufficiently peaked around the MAP estimator, then the linear approximation can be sufficiently accurate.

4.2.4 The infinite-dimensional case

In the finite-dimensional case, where the probability measures are specified by their densities with respect to the Lebesgue measure, Eq. (4.7) can be rephrased

by saying that ρ is the Radon-Nikodym derivative of the probability measure $\mu^{\mathcal{G}}$ with respect to μ_0, *viz.*

$$\frac{d\mu^{\mathcal{G}}}{d\mu^0}(u) \propto \rho(y - \mathcal{G}(u)) \,. \tag{4.65}$$

Using the fact that the density ρ is a positive function, we can rewrite

$$\rho(y - \mathcal{G}(u)) = \exp\left(-\Phi(u; y)\right) \,, \tag{4.66}$$

and therefore

$$\frac{d\mu^{\mathcal{G}}}{d\mu^0}(u) \propto \exp\left(-\Phi(u; y)\right) \,. \tag{4.67}$$

In finite-dimensional spaces, the three equations above are just definitions that do not add much content. Their interest resides in the fact that the last expression, Eq. (4.67), can be properly defined when X is infinite-dimensional, allowing a rigorous extension of the Bayesian formulation of inverse problems to the case of infinite-dimensional spaces.

Summarising the details of probability measure in infinite-dimensional spaces, is beyond the scope of this work. Adopting instead a heuristic approach, we can say that a function f is a random function if $f(x)$ is a random variable for all values of x. Since the values of the function at different values of x can be correlated, a random function is fully characterised by specifying the joint probability densities

$$\pi\left(f_1, \ldots, f_n; x_1, \ldots x_n\right) \,, \tag{4.68}$$

where $f_i = f(x_i)$, for all values of n, and all values of $x_1, \ldots, x_n$. These finite-dimensional densities allow the definition of a probability measure.

For a Gaussian random function, these densities only depend on a mean value $m(x)$ and a covariance $C(x, x')$. The probability densities for the variables f_i, for any value of n is

$$\pi\left(f_1, \ldots, f_n; x_1, \ldots, x_n\right) \propto \exp\left[-\frac{1}{2}\sum_{ij}\left(f_i - m_i\right) C^{-1}(x_i, x_j)\left(f_j - m_j\right)\right] \,. \tag{4.69}$$

The covariance C is such that

$$C(x,x') = \int df\, df'\ (f - m(x))\,(f' - m(x'))\,\pi\,(f,f';x,x')\ , \tag{4.70}$$

which shows that the two-point probability density determines all the other distributions.

4.3 NNPDF Monte Carlo approach to inverse problems

In this section we discuss the NNPDF approach to inverse problems, trying to make contact explicitly with the formalism laid out in Sec. 4.2. In the Bayesian formulation, Eq. (4.21) gives a quantitative description of how the information contained in the experimental data propagates into our knowledge of the space of models. In practice, it should be possible to sample directly from the posterior distribution. However, sampling from the posterior distribution is not straightforward and we defer investigation of this issue to further, dedicated studies. Here we focus instead on the standard NNPDF fitting procedure and investigate its relation with the Bayesian result. The NNPDF approach generates an ensemble of fit results, which are supposed to describe the posterior probability distribution for the model (*i.e.* in the space of PDFs) given the experimental data. In the case of a linear map, we show here that this is exactly the case: the NNPDF replicas are distributed exactly according to the posterior density that was obtained in the previous section.

4.3.1 Fitting replicas

The approach for generating a sample in model space employed by NNPDF can broadly be described as fitting model replicas to pseudo-data replicas. As discussed in Eq. (4.5) the experimental values are subject to observational noise. If we assume this observational noise to be multigaussian then the experimental central values, y_0, are given explicitly by

$$y_0 = f + \eta, \tag{4.71}$$

Figure 4.1 *Histogram showing the distribution of* 10^4 *replicas generated around an experimental value* y_0 *with unit variance. The central value* y_0*, which is represented by the solid dot at the centre of the replica distribution, is drawn from a Gaussian distribution with unit variance centred at the true value* f*, which is assumed to be the origin in this plot.*

where f is the vector of *true* observable values, and the observational noise is drawn from a Gaussian centered on zero such as in Eq. 4.20, *i.e.* $\eta \sim \mathcal{N}(0, C_D)$ where C_D is the experimental covariance matrix. In Eq. (4.71), each basis vector corresponds to a separate data point, and the vector of shifts η permits correlations between data points according to the covariance matrix provided by the experiments. Given the data, the NNPDF approach is to compute a MAP estimator similar to that discussed in the previous section, *i.e.* finding the model that minimises the χ^2 to the data. The key difference between the NNPDF approach and the classical MAP estimator is that instead of fitting the observational data given by Eq. 4.71, an ensemble of model replicas are fitted each to an independently sampled instance of pseudo-data, which is generated by augmenting y_0 with some noise, $\epsilon^{(k)}$,

$$\mu^{(k)} = y_0 + \epsilon^{(k)} = f + \eta + \epsilon^{(k)}, \tag{4.72}$$

where k is the replica index and each instance of the noise, ϵ, is drawn independently from the same Gaussian from which the observational noise is drawn from, *i.e.* $\epsilon \sim \mathcal{N}(0, C_D)$. A simple one-dimensional example is shown in Fig. 4.1. Note that, if we were to repeat this construction multiple times, the true value f would be within a 1σ interval centred at y_0 with a 68% probability.

The parameters for each model replica maximise the likelihood evaluated on the corresponding pseudo-data. We can think of this approach as a special case of MAP estimation, as described in Eq. (4.23), where there is no model prior that regulates the likelihood. Another way of viewing this is to take $C_M^{-1} \to 0$ in Eq. (4.23), as was done to obtain the result in Eq. 4.45. Either way, there is no

prior information about the model. The parametrization of the model is fixed, so the model space is the finite space of parameters $u \in \mathbb{R}^{N_{\text{model}}}$. In $\mathbb{R}^{N_{\text{model}}}$, we find the parameters which minimise the χ^2 between the predictions from the model and the corresponding pseudo-data $\mu^{(k)}$

$$\begin{aligned} u_*^{(k)} &= \arg\min_{u^{(k)}} \chi^{2(k)} \\ &= \arg\min_{u^{(k)}} \sum_{ij} \left(\mathcal{G}(u^{(k)}) - \mu^{(k)}\right)^T C_D^{-1} \left(\mathcal{G}(u^{(k)}) - \mu^{(k)}\right) , \end{aligned} \tag{4.73}$$

where, as usual, minimising the χ^2 is equivalent to maximising the likelihood, $\mathcal{L}$, since $\chi^2 \equiv -\log \mathcal{L}$.

As a final note: since we do not include the model prior, overall normalizations can be omitted in Eq. 4.73. It is clear however that if we were including a model prior in our MAP, it is important that the relative normalizations between the likelihood function and the model prior are self-consistent.

4.3.2 Fluctuations of fitted values

It is not immediately obvious that our MC methodology, maximising the likelihood on an ensemble of pseudo-data replicas, should guarantee that the model replicas are indeed sampled from the posterior distribution of parameters given data as described *e.g.* in Eq. 4.21. In order to investigate this issue, we will again consider a model, whose predictions are linear in the model parameters, where the posterior distribution of model parameters can be written explicitly. A practical example, which can elucidate the following arguments would be a polynomial model. Then u is a vector of N_{model} polynomial coefficients and $\mathcal{G}$ is the Vandermonde matrix

$$\mathcal{G} = \begin{bmatrix} 1 & x_1 & \dots & x_1^{N_{\text{model}}-1} \\ 1 & x_2 & \dots & x_2^{N_{\text{model}}-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{N_{\text{data}}} & \dots & x_{N_{\text{data}}}^{N_{\text{model}}-1} \end{bmatrix} . \tag{4.74}$$

In this case the forward map yields

$$y_i = \sum_{j=0}^{N_{\text{model}}-1} u_j x_i^j , \tag{4.75}$$

where $i = 1, \ldots, N_{\text{data}}$. The arguments here are not restricted to polynomials, however, and apply to any model whose forward map can be expressed as Eq. (4.40), for example a linear shallow approximation of neural networks [145]. In order to get an exact analytical solution for the linear model, we additionally require $\mathcal{G}$ to have linearly independent rows, and therefore $\mathcal{G}C_D\mathcal{G}^T$ is invertible. With no prior information on the model, the posterior distribution of model parameters is a Gaussian with mean and covariance given by Eqs. 4.45 and 4.46.

If instead we deploy the NNPDF Monte Carlo method to fitting model replicas, then in the case under study $\arg\min_{u^{(k)}} \chi^{2(k)}$ is found analytically by imposing thatthe derivative of $\chi^{2(k)}$ with respect to the model parameters is zero, i.e.

$$u_*^{(k)} = (\mathcal{G}^T C_D^{-1} \mathcal{G})^{-1} \left(\mathcal{G}^T C_D^{-1} y_0 + \mathcal{G}^T C_D^{-1} \epsilon^{(k)} \right) . \tag{4.76}$$

Eq. 4.76 shows that u_* is a linear combination of the Gaussian variables ϵ, and therefore is also a Gaussian variable. Its probability density is then completely specified by the average and covariance of u_*, which can be calculated explicitly, given that the probability density for ϵ is known:

$$\mathbf{E}_{\{u_*\}} [u_*] = \tilde{u} = (\mathcal{G}^T C_D^{-1} \mathcal{G})^{-1} \mathcal{G}^T C_D^{-1} y_0 \tag{4.77}$$

$$\text{cov}(u_*) = \tilde{C}_M = (\mathcal{G}^T C_D^{-1} \mathcal{G})^{-1} . \tag{4.78}$$

We see explicitly here that, under the assumptions specified above, $u_* \sim \mathcal{N}(\tilde{u}, \tilde{C}_M)$. In other words, when the model predictions are linear in the model parameters, the NNPDF MC method is shown to produce a sample of models from the posterior distribution of model parameters given the data. When we fit PDFs, parametrized as deep fully connected neural networks, to data which includes hadronic observables, it is clear that the forward map is non-linear, and therefore this proof does not strictly apply. As previously discussed at the end of Sec. 4.2.3, even for non-linear models we can make a linear approximation of the forward map provided that we are expanding around the MAP estimator. This means the NNPDF MC methodology should reproduce the posterior distribution of the model given the data, at least close $\tilde{u}$, the central value of the fitted replicas. Furthermore, by fluctuating the data and fitting the replicas, the fluctuations in data space are propagated to model space non-linearly. So even for non-linear problems, the NNPDF MC methodology will produce a sample of models which are at least approximately distributed according to the posterior model distribution. It remains to be shown, however, that further away from the MAP estimator the

approximation holds despite the non-linear dependence of the model replicas on the data uncertainties.

4.3.3 Closure test

The concept of the closure test, which was first introduced in Ref. [2], is to construct artificial data by using a known pre-existing function to generate the *true* observable values, f. One way of achieving this is by choosing w such that $f = \mathcal{G}(w)$. Then the experimental central values are artificially generated according to Eq. 4.71, where the observational noise is pseudo-randomly generated from the assumed distribution.

In [2], f is referred to as level 0 (L0) data and y_0 is referred to as level 1 (L1) data. Finally, if we use the NNPDF MC method to fit artificially generated closure data, the pseudo-data replicas that are fitted by the model replicas are referred to as level 2 (L2) data.

In a closure test, the assumed prior of the data is fully consistent with the particular instance of observed central values, y_0, by construction. In the original closure test in NNDPF3.0 there was also no modelisation uncertainty, the true observable values were assumed to be obtained by applying the forward map $\mathcal{G}$ to a vector in model space w. It is worth noting that the assumption of zero modelisation uncertainties is quite strong and likely unjustified in many areas of physics. In the context of fitting parton distribution functions there are potentially missing higher order uncertainties (MHOUs) from using fixed order perturbative calculations as part of the forward map. MHOUs have been included in parton distribution fits [146] and in the future these should be included in the closure test, however this is beyond the scope of the study presented here, since MHOUs are still not included in the NNPDF methodology by default. In the results presented in the rest of this paper we do include nuclear and deuteron uncertainties, as presented in [147, 148], since they are to be included in NNPDF fits by default. Extensive details for including theoretical uncertainties, modelled as theoretical covariance matrices can be found in those references. For the purpose of this study the modelisation uncertainty is absorbed into the prior of the data, since

$$y_0 = \mathcal{G}(u) + \eta + \delta \tag{4.79}$$

where $\delta \sim \mathcal{N}(0, C^{\text{theory}})$. As long as the modelisation uncertainty is independent

of the data uncertainty, we can absorb δ into η by modifying the data prior: $\eta \sim \mathcal{N}(0, C + C^{\text{theory}})$. In doing that, we must also update the likelihood of the data given the model to use the total covariance $(C + C^{\text{theory}})$. From now onwards we will omit C^{theory} because it is implicit that we always sample and fit data using the total covariance matrix which includes any modelisation uncertainty we currently take into account as part of our methodology.

Mapping the closure test procedure to the quantities used in the Bayesian treatment presented in the previous section will allow us to derive a number of analytical results in Sect. 4.4.4.

4.4 Data space estimators

In order to perform a quantitative analysis of the results obtained in the closure tests, we discuss several estimators, which are computed from the outcome of the closure test fits. These results depend on the pseudo-data that have been generated and therefore are stochastic variables which can fluctuate. The values of the estimators on a single replica will not tell anything about the quality of our fits: we need to understand their probability distributions in order to validate our fitting procedure. We begin this section by defining estimators in data space, *i.e.* estimators that are computed from the model predictions for a set of experimental data points. Having defined the estimators, we define criteria to characterise faithful uncertainties. We conclude this section with a discussion of the predictions that can be obtained for these estimators in the case of a linear model, where analytical calculations can be performed. As we already discussed, the analytical results cannot be applied directly to the NNPDF fits, but they are useful examples that illustrate the expected behaviour of these quantities.

4.4.1 Deriving the data space estimators

For a given model $u_*^{(k)}$, obtained from fitting the k-th replica, we start by defining the model error as the χ^2 between the model predictions and some data central values ${y_0}'$, normalized by the number of data points

$$\frac{1}{N_{\text{data}}} \left(\mathcal{G}'\left(u_*^{(k)}\right) - {y_0}'\right)^T {C_D}'^{-1} \left(\mathcal{G}'\left(u_*^{(k)}\right) - {y_0}'\right) , \tag{4.80}$$

and we purposely denoted the data which the model error is evaluated on as ${y_0}'$, as opposed to the data which is used to determine the model parameters y_0. Note that in Eq. 4.80, y_0' is a stochastic variable, but also $u_*^{(k)}$ is a stochastic variable, with its pattern of fluctuations, since the fitted model depends on the data $\mu^{(k)}$ that enter the fit. We define the model error $\mathcal{E}^{\text{out}}$ on the set of data ${y_0}'$ by taking the average over the models,

$$\mathcal{E}^{\text{out}} = \frac{1}{N_{\text{data}}} \mathbf{E}_{\{u_*\}} \left[\left(\mathcal{G}' \left(u_*^{(k)} \right) - {y_0}' \right)^T {C_D}'^{-1} \left(\mathcal{G}' \left(u_*^{(k)} \right) - {y_0}' \right) \right] , \tag{4.81}$$

where we defined the expectation value over the ensemble of model replicas as

$$\mathbf{E}_{\{u_*\}} [x] \equiv \frac{1}{N_{\text{replicas}}} \sum_{k=1}^{N_{\text{replicas}}} x^{(k)} . \tag{4.82}$$

We could of course set ${y_0}' = y_0$ and evaluate the model performance on the fitted data however, as is common in machine learning literature, we intend to use a separate set of test data. Ideally we would choose ${y_0}'$ such that ${y_0}'$ and y_0 are statistically independent, as in Eq. 4.52. This is achieved by choosing the split such that the experimental covariance matrix is block diagonal:

$$C_D^{\text{total}} = \begin{bmatrix} C_D & 0 \\ 0 & {C_D}' \end{bmatrix} . \tag{4.83}$$

It is useful to perform a decomposition of Eq. 4.81, following usual manipulations of the likelihood function associated with least-squares regression in [13]. Least-squares regression is a special case of minimum likelihood estimation, where the uncertainty on each data point is equal in magnitude and uncorrelated. Here we review the decomposition in the more general framework of data whose uncertainty is multigaussian. Starting with Eq. 4.81 (evaluated on the ideal test data), we can complete the square

$$\begin{aligned} \mathcal{E}^{\text{out}} = & \frac{1}{N_{\text{data}}} \mathbf{E}_{\{u_*\}} \left[\left(\mathcal{G}' \left(u_*^{(k)} \right) - f' \right)^T {C_D}'^{-1} \left(\mathcal{G}' \left(u_*^{(k)} \right) - f' \right) \right] + \\ & + \mathbf{E}_{\{u_*\}} \left[\left(f' - {y_0}' \right)^T {C_D}'^{-1} \left(f' - {y_0}' \right) \right] + \\ & + 2\mathbf{E}_{\{u_*\}} \left[\left(\mathcal{G}' \left(u_*^{(k)} \right) - f' \right)^T {C_D}'^{-1} \left(f' - {y_0}' \right) \right] . \end{aligned} \tag{4.84}$$

The second term is the shift associated with evaluating the model error on noisey test data and the final term is a cross term which we will deal with later. For now

we focus on decomposing the first term further,

$$\begin{aligned}
&\mathbf{E}_{\{u_*\}}\left[\left(\mathcal{G}'\left(u_*^{(k)}\right)-f'\right)^T {C_D}'^{-1}\left(\mathcal{G}'\left(u_*^{(k)}\right)-f'\right)\right]= \\
&=\mathbf{E}_{\{u_*\}}\left[\left(\mathcal{G}'\left(u_*^{(k)}\right)-\mathbf{E}_{\{u_*\}}\left[\mathcal{G}'\left(u_*^{(k)}\right)\right]\right)^T {C_D}'^{-1}\left(\mathcal{G}'\left(u_*^{(k)}\right)-\mathbf{E}_{\{u_*\}}\left[\mathcal{G}'\left(u_*^{(k)}\right)\right]\right)\right]+ \\
&+\left(\mathbf{E}_{\{u_*\}}\left[\mathcal{G}'\left(u_*^{(k)}\right)\right]-f'\right)^T {C_D}'^{-1}\left(\mathbf{E}_{\{u_*\}}\left[\mathcal{G}'\left(u_*^{(k)}\right)\right]-f'\right)\,,
\end{aligned} \tag{4.85}$$

where we have used the fact that the second term is constant across replicas and the cross term that arises in this decomposition is zero when the expectation value across replicas is taken. The first term in this expression we call the *variance* and the second term is the *bias*.

As previously mentioned $\mathcal{E}^{\text{out}}$ should be considered a stochastic estimator, in theory we could take the expectation value across training data y_0 and test data ${y_0}'$, the latter of which cancels the cross term in Eq. 4.84. The final result of that would be

$$\mathbf{E}_{y_0,{y_0}'}[\mathcal{E}^{\text{out}}]=\mathbf{E}_{y_0}[\text{bias}]+\mathbf{E}_{y_0}[\text{variance}]+\mathbf{E}_{{y_0}'}[\text{noise}]\,. \tag{4.86}$$

We are not interested in the observational noise term, since it is independent of the model and in the limit of infinite test data $\mathbf{E}_{{y_0}'}[\text{noise}]\to 1$. The two estimators of interest are independent of the test data, and therefore we only need to take the expectation value over the training data.

Multiple closure fits In practical terms, taking the expectation value across the training data can be achieved by running multiple closure fits, each with a different observational noise vector η, and taking the average i.e.

$$\mathbf{E}_{y_0}[x]=\frac{1}{N_{\text{fits}}}\sum_{j=1}^{N_{\text{fits}}}x. \tag{4.87}$$

Clearly this is resource intensive, and requires us to perform many fits. In NNPDF3.0 [2], single replica proxy fits were used to perform a study of the uncertainties. Here we have expanded the data-space estimators used in the closure fits and also will be using multiple full replica fits to calculate various expectation values - made possible by our next generation fitting code [7].

4.4.2 Geometric Interpretation

It is possible to interpret the relevant data space estimators geometrically, by considering a coordinate system where each basis vector corresponds to an eigenvector of the experimental covariance matrix normalized by the square root of the corresponding eigenvalue. An example of this is given in Fig. 4.2, where for simplicity we have considered a system with just two data points, *i.e.* a two-dimensional data space, with a diagonal covariance.

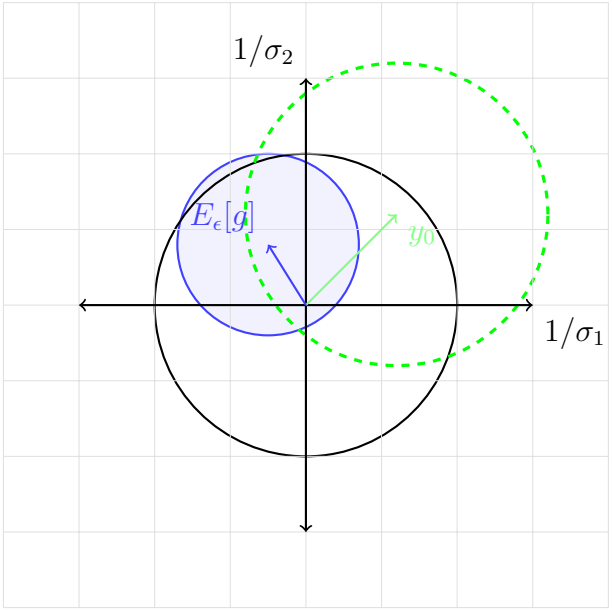

Figure 4.2 *Example of geometric interpretation of closure test estimators. The origin is the true observable values for each data point. The level one data (or experimental central values) are shifted away from this by η. In this example the covariance matrix is diagonal, so the eigenvectors correspond to the two data points, the square root of the eigenvalues are simply the standard deviation of those points. This is without loss of generality because any multivariate distribution can be rotated into a basis which diagonalises the covariance matrix. The 1-sigma observational noise confidence interval is a unit circle centered on the origin. Some closure estimators can be understood as l2-norms of the vectors connecting points, i.e the bias is the l2-norm of the vector from the origin to the central value of the predictions.*

The origin of the coordinate system is the true value of the observable. The observational noise in these coordinates corresponds to a unit circle centred in the origin as shown in Fig. 4.2. If the experimental covariance is faithful, there is a 68% probability that the experimental value y_0 is within this unit circle. Fig. 4.2 shows one possible instance of y_0. Repeating the entire fit procedure multiple

times requires generating new sets of experimental data y_0. The average over y_0 mentioned above, is precisely the average over multiple fits, restarting the procedure each time from a new instance of y_0.

For a given y_0 the replicas are generated as a set of points Gaussianly distributed around it and therefore, in the limit of a large number of replicas, 68% of them will fall within a unit circle centred in y_0. This is the dashed circle in the figure. Clearly there is also a 68% probability that the true value (*i.e.* the origin in our plot) is inside this second circle. The model predictions, one for each replica, are then a set of points, whose mean is $E_\epsilon[g]$. The mean squared radius of those points is what we call the variance. The bias is the l2-norm of the vector between the origin and the mean of the model predictions.

A faithful representation of the errors requires that the true value, *i.e.* the origin of the coordinate system in our figure, has 68% probability of being within 1σ from the central value of the fit, which is given by $E_\epsilon[g]$. Looking at the figure again, the probability for the origin to be inside the shaded circle must be 68%. We will discuss faithful errors in more detail in the next subsection.

4.4.3 Faithful uncertainties in data space

The two closure estimators of interest, bias and variance, can be used to understand faithful uncertainties in a practical sense. If we return to Eq. 4.86 we can examine both estimators in a bit more detail.

Variance The *variance* in the above decomposition refers to the variance of the model predictions in units of the covariance

$$\begin{aligned}\text{variance} = \frac{1}{N_{\text{data}}}\mathbf{E}_{\{u_*\}}\Big[\\ \left(\mathcal{G}'\left(u_*^{(k)}\right) - \mathbf{E}_{\{u_*\}}\left[\mathcal{G}'\left(u_*^{(k)}\right)\right]\right)^T {C_D}'^{-1} \left(\mathcal{G}'\left(u_*^{(k)}\right) - \mathbf{E}_{\{u_*\}}\left[\mathcal{G}'\left(u_*^{(k)}\right)\right]\right)\Big],\end{aligned} \tag{4.88}$$

which can be interpreted as the model uncertainty in the space of the test data. It is instructive to rephrase Eq. 4.88 as

$$\text{variance} = \frac{1}{N_{\text{data}}}\text{Tr}\left[{C'}^{(\text{replica})}{C_D}'^{-1}\right], \tag{4.89}$$

where

$$C'^{(\text{replica})} = \mathbf{E}_{\{u_*\}}\left[\left(\mathcal{G}'\left(u_*^{(k)}\right) - \mathbf{E}_{\{u_*\}}\left[\mathcal{G}'\left(u_*^{(k)}\right)\right]\right)\left(\mathcal{G}'\left(u_*^{(k)}\right) - \mathbf{E}_{\{u_*\}}\left[\mathcal{G}'\left(u_*^{(k)}\right)\right]\right)^T\right] \tag{4.90}$$

is the covariance matrix of the predictions from the model replicas. Note that we can rotate to a basis where ${C_D}'$ is diagonal,

$$\left({C_D}'^{-1}\right)_{ij} = \frac{1}{\left(\sigma'_i\right)^2}\delta_{ij}\,, \tag{4.91}$$

then we can rewrite Eq. 4.89 as

$$\text{variance} = \frac{1}{N_{\text{data}}}\sum_i \frac{C'^{(\text{replica})}_{ii}}{\left(\sigma'_i\right)^2}\,. \tag{4.92}$$

The numerator in the right-hand side of the equation above is the variance of the theoretical prediction obtained from the fitted replicas, while the denominator is the experimental variance, the average is now taken over eigenvectors of the experimental covariance matrix.

Bias Similarly, the *bias* is defined as the difference between the expectation value of the model predictions and the true observable values in units of the covariance, *i.e.*

$$\text{bias} = \frac{1}{N_{\text{data}}}\left(\mathbf{E}_{\{u_*\}}\left[\mathcal{G}'\left(u_*^{(k)}\right)\right] - f'\right)^T {C_D}'^{-1}\left(\mathbf{E}_{\{u_*\}}\left[\mathcal{G}'\left(u_*^{(k)}\right)\right] - f'\right)\,. \tag{4.93}$$

The smaller the bias, the closer the central value of the predictions is to the underlying law. In Eq. 4.86, the expectation value is taken across the prior distribution of the training data, which yields

$$\mathbf{E}_{y_0}[\text{bias}] = \frac{1}{N_{\text{data}}}\text{Tr}\left[C'^{(\text{central})}{C_D}'^{-1}\right]\,, \tag{4.94}$$

where we have introduced $C'^{(\text{central})}$ as the covariance of the expectation value of the model predictions,

$$C'^{(\text{central})} = \mathbf{E}_{y_0}\left[\left(\mathbf{E}_{\{u_*\}}\left[\mathcal{G}'\left(u_*^{(k)}\right)\right] - f'\right)\left(\mathbf{E}_{\{u_*\}}\left[\mathcal{G}'\left(u_*^{(k)}\right)\right] - f'\right)^T\right]\,. \tag{4.95}$$

The point is that the bias on the test data is a stochastic variable which depends on the central value of the training data through $u_*^{(k)}$. The matrix $C'^{(\text{central})}$

describes the fluctuations of the central value of the model prediction around the true observable values as we scan different realisations of the training data.

It is important to stress the difference between variance and bias. In the case of the variance, we are looking at the fluctuations of the replicas around their central value for fixed y_0. This is related the ensemble of model replicas we provide as the end product of a fit and can be calculated when we have one instance of y_0, provided by the experiments. In the case of the bias we consider the fluctuations of the central value over replicas around the true theoretical prediction as the values of y_0 fluctuate around f. This latter procedure is only possible in a closure test, where the underlying true observable is known. The bias as defined here yields an estimate of the fluctuations of the MAP estimator if we could do multiple independent experiments.

Bias-variance ratio Finally, the *bias-variance ratio* is defined as

$$\mathcal{R}_{bv} \equiv \sqrt{\frac{\mathbf{E}_{y_0}[\text{bias}]}{\mathbf{E}_{y_0}[\text{variance}]}}\,, \tag{4.96}$$

where we have taken the square root, since bias and variance are both mean squared quantities. The value of $\mathcal{R}_{bv}$ yields a measurement of how much uncertainties are over or under estimated. If the uncertainties are completely faithful, then $\mathcal{R}_{bv} = 1$. We note that the relationship does not work both ways and $\mathcal{R}_{bv} = 1$ does not necessarily guarantee that the uncertainty is faithful. We also note that $\mathcal{R}_{bv}$ is not completely general: it is not a measure defined in model space and depends on the choice of test data. Therefore it only gives *local* information on the model uncertainties. If the distribution of the expectation value of model predictions is gaussian centered on the true observable values, with covariance $C'^{(\text{central})}$ and the distribution of the model replicas is also gaussian, with covariance $C'^{(\text{replica})}$ then model uncertainties are faithful if

$$C'^{(\text{central})} C'^{(\text{replica})^{-1}} = 1. \tag{4.97}$$

The difficulty with calculating Eq. 4.97 comes from the fact that $C'^{(\text{replica})}$ is likely to have large correlations which would lead it to be singular or ill-conditioned. As a result, any error estimating $C'^{(\text{replica})}$ from finite number of replicas could lead to unstable results. $\mathcal{R}_{bv}$ overcomes this instability by taking the ratio of the average across test data of these matrices, in units of the experimental covariance matrix.

There may still be large relative errors for smaller eigenvalues of $C'^{(\text{replica})}$, but these should not lead to instabilities in $\mathcal{R}_{bv}$ unless they correspond to directions with very low experimental uncertainty. As an extra precaution, we shall estimate an uncertainty on $\mathcal{R}_{bv}$ by performing a bootstrap sample on fits and replicas.

Quantile statistics When the closure test was first presented in [2], there was an estimator introduced in the space of PDFs which also aimed to estimate faithfulness of PDF uncertainties using the combined assumption of Gaussian PDF uncertainties and quantile statistics, called $\xi_{1\sigma}$. Here we can define an analogous expression in the space of data,

$$\xi^{(\text{data})}_{n\sigma'} = \frac{1}{N_{\text{data}}} \sum_{i}^{N_{\text{data}}} \frac{1}{N_{\text{fits}}} \sum_{l}^{N_{\text{fits}}} I_{[-n\sigma'^{(l)}_i, n\sigma'^{(l)}_i]} \left(\mathbf{E}_{\{u_*\}} \left[\mathcal{G}'_i \right]^{(l)} - f'_i \right), \tag{4.98}$$

where $\sigma'^{(l)}_i = \sqrt{C'^{(\text{replica})}_{ii}}$ is the standard deviation of the theory predictions estimated from the replicas of fit l and $I_{[a,b]}(x)$ is the indicator function, which is one when $a \leq x \leq b$ and zero otherwise. In other words, $\xi^{(\text{data})}_{n\sigma'}$ is counting how often the difference between the prediction from the MAP estimator and the true observable value is within the $n\sigma$-confidence interval of the replicas, assuming they're Gaussian. Since $C'^{(\text{replica})}$ is primarily driven by the replica fluctuations, we assume that it is roughly constant across fits, or independent upon the specific instance of observational noise. This allows us to write $\xi^{(\text{data})}_{n\sigma'}$ for a specific data point in the limit of infinite fits, each to a different instance of the data as

$$\xi^{(\text{data})}_{n\sigma'_i} = \int_{-\infty}^{\infty} I_{[-n\sigma'_i, n\sigma'_i]} \left(\mathbf{E}_{\{u_*\}} \left[\mathcal{G}'_i \right]^{(l)} - f'_i \right) \rho(\eta) \, \mathrm{d}(\eta) \,, \tag{4.99}$$

where $\mathbf{E}_{\{u_*\}} \left[\mathcal{G}_i \right]^{(l)}$ has implicit conditional dependence on η. If the distribution of $\mathbf{E}_{\{u_*\}} \left[\mathcal{G}'_i \right]^{(l)} - f'_i$ is Gaussian, centered on zero, we can defined $\hat{\sigma}'_i = \sqrt{C'^{(\text{central})}_{ii}}$. In which case

$$\xi^{(\text{data})}_{n\sigma'_i} = \text{erf} \left(\frac{n\sigma'_i}{\hat{\sigma}'_i \sqrt{2}} \right), \tag{4.100}$$

which is simply the standard result of integrating a gaussian over some finite symmetric interval.

The analogy between $\mathcal{R}_{bv}$ and $\xi^{(\text{data})}_{n\sigma'}$ is clear, the ratios of uncertainties are both attempts to quantify Eq. 4.97 whilst keeping effects due to using finite statistics under control. Whilst with $\mathcal{R}_{bv}$ we take the average over test data before taking the

ratio, $\xi^{(\text{data})}_{n\sigma'}$ instead takes the ratio of the diagonal elements - ignoring correlations. Since the predictions from the model will be compared with experimental central values, taking into account experimental error, we find it more natural to calculate $\xi^{(\text{data})}_{n\sigma'}$ in the basis which diagonalises the experimental covariance of the test data as in Eq. 4.91. If we assume that in this new basis, that both $\frac{C'^{(\text{replica})}_{ii}}{\left(\sigma'_i\right)^2}$ and $\frac{C'^{(\text{central})}_{ii}}{\left(\sigma'_i\right)^2}$ are approximately constant for all eigenvectors of the experimental covariance matrix, then we recover the approximation

$$\xi^{(\text{data})}_{n\sigma'} \sim \text{erf}\left(\frac{n\mathcal{R}_{bv}}{\sqrt{2}}\right). \tag{4.101}$$

Whilst it is clear that Eq. 4.101 is reliant on a fair few assumptions which may not hold, we will use the comparison of $\xi^{(\text{data})}_{n\sigma'}$ with $\mathcal{R}_{bv}$ to consider how valid these assumptions may be.

4.4.4 Closure estimators - Linear problems

Once again we return to the linear model framework set out in Sec. 4.3.2. We can perform an analytical closure test in this framework, and check our understanding of the closure estimators. Consider the true observable values for the test data is given by

$$f' = \mathcal{G}' w \tag{4.102}$$

where $w \in X$, which means the number of (non-zero) parameters in the underlying law is less than or equal to the number of parameters in the model, $N_{\text{law}} \leq N_{\text{model}}$. Using the previous results from Sec. 4.3.2, we can write down the difference between the true observables and the predictions from the MAP estimator (or the expectation of the model predictions across model replicas - in the linear model these are the same)

$$\begin{aligned} \mathbf{E}_{\{u_*\}}\left[\mathcal{G}'\left(u^{(k)}_*\right)\right] - f' &= \mathcal{G}'(\tilde{u} - w) \\ &= \mathcal{G}'\tilde{C}_M\mathcal{G}^T C_D\,\eta\,, \end{aligned} \tag{4.103}$$

where we recall that $\mathcal{G}$ is the forward map to the training observables and y_0 are the corresponding training central values. Calculating the covariance across training data of $\mathbf{E}_{\{u_*\}}\left[\mathcal{G}'\left(u^{(k)}_*\right)\right] - f'$ gives

$$C'^{(\text{central})} = \mathcal{G}'\tilde{C}_M\mathcal{G}'^T\,, \tag{4.104}$$

so the full expression for $\mathbf{E}_{y_0}[\text{bias}]$ is given by

$$\mathbf{E}_{y_0}[\text{bias}] = \frac{1}{N_{\text{data}}}\text{Tr}\left[\mathcal{G}'\tilde{C}_M\mathcal{G}'^T{C_D}'^{-1}\right]. \tag{4.105}$$

We note that if the test data is identical the data the model was fitted on, we recover an intuitive result $\mathbf{E}_{y_0}[\text{bias}] = \frac{N_{\text{model}}}{N_{\text{data}}}$. Consider the example of the polynomial, the maximum value which N_{model} can take whilst $\mathcal{G}$ still has linearly independent rows is N_{data} and in this case the $\mathbf{E}_{y_0}[\text{bias}]$ takes its maximum value of 1. The central predictions from the model exactly pass through each data point.

We can perform a similar exercise on the model replica predictions. The difference between the predictions from model replica (k) and the expectation value of the model predictions is

$$\begin{aligned}\mathcal{G}'\left(u_*^{(k)}\right) - \mathbf{E}_{\{u_*\}}\left[\mathcal{G}'\left(u_*^{(k)}\right)\right] &= \mathcal{G}'(u_*^{(k)} - \tilde{u}) \\ &= \mathcal{G}'\tilde{C}_M\mathcal{G}^T C_D\,\epsilon\,.\end{aligned} \tag{4.106}$$

Since ϵ and η follow the same distribution, it is clear that

$$C'^{(\text{replica})} = C'^{(\text{central})}, \tag{4.107}$$

which, as a result means that

$$\text{variance} = \mathbf{E}_{y_0}[\text{bias}]. \tag{4.108}$$

We recall that when the map is linear, the NNPDF MC methodology generates replicas which are sampled from the posterior distribution of the model given the data. We have shown here that provided the underlying law belongs to the model space, the posterior distribution of the model predictions satisfy the requirement that $\mathcal{R}_{bv} = 1$.

We note that due to the invariance of the trace under cyclic permutations, we can rearrange Eq. 4.105 as

$$\mathbf{E}_{y_0}[\text{bias}] = \frac{1}{N_{\text{data}}}\text{Tr}\left[\tilde{C}_M\mathcal{G}'^T{C_D}'^{-1}\mathcal{G}'\right], \tag{4.109}$$

where the term $\mathcal{G}'^T{C_D}'^{-1}\mathcal{G}'$ can be understood as the covariance matrix of the posterior distribution in model space given the test data, with zero prior knowledge

of the model *viz.*

$$\mathbf{E}_{y_0}[\text{bias}] = \frac{1}{N_{\text{data}}}\text{Tr}\left[\tilde{C}_M \tilde{C}'^{-1}_M\right] , \tag{4.110}$$

where we emphasise that the covariance matrices $\tilde{C}_M$ are $\tilde{C}'_M$ from completely independent Bayesian inferences with no prior information on the model parameters, unlike in Eq. 4.59 where a sequential marginalisation causes $\tilde{C}'_M$ to depend on $\tilde{C}_M$.

Alternatively, if we perform a sequential marginalisation of the data, and use the result in Eq. 4.59, but then take ${C_D}'^{-1} \to 0$, *i.e.* there is no information on the observables in the test set, then

$$\tilde{C}_M^{-1} = \mathcal{G}^T C_D^{-1} \mathcal{G} , \tag{4.111}$$

or the total posterior model distribution, is identical to the posterior model distribution given just the training data - as you would expect. Now we can express bias (or variance) as

$$\mathbf{E}_{y_0}[\text{bias}] = \frac{1}{N_{\text{data}}}\text{Tr}\left[\tilde{C}'_D {C_D}'^{-1}\right] , \tag{4.112}$$

where $\tilde{C}'_D$ is the covariance of the posterior distribution of y' with no prior information on that data. This might seem peculiar because in determining $\tilde{C}'_D$ we took the limit ${C_D}'^{-1} \to 0$, because we had no prior information on the unseen data, however in Eq. 4.112 we require ${C_D}'^{-1}$ to be finite. We rationalise Eq. 4.112 as a comparison between the posterior distribution in the space of data of some unseen observables to an independently determined prior from performing the relevant experiment which measures the same observables. Comparing moments of these two distributions is what you would expect when the new experimental data is published.

Underparametrized model Note that if we were to choose the number of model parameters such that $N_{\text{law}} > N_{\text{model}}$, then the variance would be unaffected, since the underlying law parameters cancel. However, the bias would now contain an extra term from the extra parameters in the underlying law, schematically:

$$(\mathbf{E}_{\{u_*\}}\left[\mathcal{G}'\left(u_*^{(k)}\right)\right] - f')_i = \sum_{1\leq j\leq N_{\text{model}}} {\mathcal{G}'}_{ij}(\tilde{u} - w)_j - \sum_{N_{\text{model}}<j\leq N_{\text{law}}} {\mathcal{G}'}_{ij} w_j, \tag{4.113}$$

which would mean that $\mathcal{R}_{bv} \neq 1$. This demonstrates that requiring $\mathcal{R}_{bv} = 1$ demands that the model space is suitably flexible, if the underlying law is parametrized then this can be summarised as requiring $w \in X$. Note that in the underparametrized regime the model replicas are still drawn from the posterior distribution, however because $w \notin X$ we've somehow invalidated the assumptions that go into the relation between model predictions and the data-space prior.

Although $\mathcal{R}_{bv}$ was largely chosen on practical grounds, we see that it is still a stringent test that our assumptions are correct and that the distribution our model replicas are drawn from is meaningful, this is what we mean when we say *faithful uncertainties*.

An unfortunate trade-off when using $\mathcal{R}_{bv}$ is it can't be used as a diagnostic tool, and is instead used simply for validation. For example, if $\mathcal{R}_{bv} > 1$, then we can't know whether there was a problem with the fitting methodology used to generate the model replicas or a deeper issue such as an inflexible model.

4.5 Experimental setup

Here we discuss the experimental setup used to produce the results. The results here act mainly as a proof of principle of the data space estimators presented in this Chapter in a realistic inverse problem setting.

The estimators were used as part of a suite of methodological validation tools, see also the "future tests" [149], used to understand the PDF uncertainties of the NNPDF4.0 set of PDFs [6], and provide evidence that they are faithful. For the purpose of understanding how the results here were produced, we will briefly describe the key features of the NNPDF4.0 methodology, but refer the reader to NNPDF4.0 for a full discussion on how these methodological choices were made, and the impact on performing PDF fits to experimental data. The main point of these experimental results is that we can use the data space estimators in a setting with a highly non-linear forward map. We aim to show that even in this setting the NNPDF MC approach to inverse problems whereby we fit fluctuated data replicas is successful in producing a representative sample from the posterior distribution of the model. We therefore demonstrate both the usefulness of the estimators themselves as well as the the NNPDF MC approach to inverse problems.

4.5.1 Neural network parton distribution functions

Using neural networks to fit PDFs has been discussed many times in previous NNPDF publications, see for example [2, 150]. A new feature of NNPDF4.0 [6] is that, for the default fit, a single neural network parametrizes all 8 of the fitted PDF flavours at the initial scale Q_0. The PDF for a single flavour j, at the initial scale $Q_0 = 1.65\text{GeV}$ as a function of the momentum fraction x is given by

$$f_j(x, Q_0) = NN(x, \ln x|u)_j * x^{1-\alpha_j} * (1-x)^{\beta_j}, \tag{4.114}$$

where α and β are the pre-processing exponents, which control the PDF behaviour as $x \to 0$ and $x \to 1$ respectively and $NN(x, \ln x|u)_j$ is the j^{th} output of the neural network, which takes x and $\ln x$ as input. The pre-processing exponents ensure that the PDFs remain integrable even if there is no data to constrain the parameters at low and high x. As discussed in Sec. 4.3.1, an ensemble of models is fitted, each one is an MAP estimator of the corresponding pseudo-data it is fitted on. Unlike in the case of the linear model, the parameters of the neural network cannot be found analytically and instead an optimization algorithm is used to try and find the parameters which maximise the likelihood. In principle, the pre-processing exponents can also be varied during the fit analogously to the neural network parameters, such as in [151], or they can be randomly selected from a predetermined range as is done in previous NNPDF releases, for example [150]. There are clearly many choices with respect to hyperparameters, the full discussion of how these choices have been made is beyond the scope of this Chapter, however they were chosen as a result of performing a hyperparameter scan. The hyperparameter scan involves running multiple fits with different values for the hyperparameter and selecting the "best" set of parameters based on some criterion. Here the best parameters were chosen based on the goodness of fit (χ^2) to some held out data. For extensive details on the selection process of the hyperparameters, see the full NNPDF4.0 release [6]. A summary of the hyperparameters used to produce results presented in this paper are provided in Tab. C.2.

Finally, the parton distributions themselves are not compared directly with data. Instead the observables quantities are obtained by performing convolutions between partonic cross sections and the PDFs, as discussed in Eq. 4.2. This is a direct consequence of factorization theorem [152], which states that observables from hadronic interactions can be factorized such that the PDFs are universal

functions which contain divergences from the emission of soft and collinear gluons, and partonic cross sections are calculable from perturbative QCD. For DIS-like observables, where the proton is probed by scattering leptons off a stationary hadronic target, the observables can be formulated as a convolution with a single PDF such as in Eq. 4.2.

We note in that equation that both the observable and the PDF are functions of the momentum fraction $x \in [0, 1]$ and the energy scale Q. However, in Eq. 4.114 the PDFs themselves are only parametrized at a single scale, which in the PDF fitting jargon is called the fitting, initial or parametrization scale Q_0. When the soft and collinear divergences are absorbed into the PDF, it induces a dependence on an artificial factorization scale, referred to as μ_F. However, since the physical observables are independent of this artificial scale, we are left with a set of renormalization group equations, known as the DGLAP equations [153–155]. These are differential equations, which can be solved numerically to "evolve" PDFs from one scale to another. It is convenient to re-express the DGLAP equation in a basis of PDF flavours which maximally diagonalises them, which is achieved by parametrizing the PDFs in the so-called evolution basis, whose eight flavours, $\{g, \Sigma, V, V_3, V_8, T_3, T_8, c\}$ are defined as

- g
- $\Sigma \equiv \sum_{q_i \in \{u,d,s\}} q_i + \bar{q}_i$
- $V \equiv \sum_{q_i \in \{u,d,s\}} q_i - \bar{q}_i$
- $V_3 \equiv u - \bar{u} - (d - \bar{d})$
- $V_8 \equiv u - \bar{u} + d - \bar{d} - 2(s - \bar{s})$
- $T_3 \equiv u + \bar{u} - (d + \bar{d})$
- $T_8 \equiv u + \bar{u} + d + \bar{d} - 2(s + \bar{s})$
- c

where $\{u, d, s, c\}$ correspond to up, down, strange and charm quark PDFs, g is the gluon PDF and a bar is used to indicate the anti-quark PDF.

In practice the convolution between the partonic cross section and the PDFs can be discretized and the evolution of the PDF from the initial scale Q_0 up to the

energy scale of the observable Q, from solving the DGLAP equations, can be pre-calculated. The evolution and partonic cross section are then contained in an object, which is a core component of the NNPDF fitting methodology, known as a FastKernel table [156, 157], for each data point. A detailed discussion of FastKernel (FK) tables is given in [158]. A tensor product between the PDF at an initial scale Q_0, on a grid of points in x and the FK tables results in

- Evolution of the PDFs from the initial scale Q_0 up to the energy scale of the process.
- Linear combinations of the evolved PDFs depending on the sub-processes for the given observable.
- Convolution between the partonic cross section and the relevant linear combinations of evolved PDFs.

In practice the observables considered in these experiments define highly non-trivial forward maps, for DIS-like observables:

$$\mathcal{G}\left(u\right)_i = \sum_{\alpha}^{N_x} \sum_{l}^{N_{\text{flav}}} A_i^{\alpha l} f_l(x_\alpha, Q_0|u) \ , \tag{4.115}$$

where $A_i^{\alpha l}$ is the FK table for DIS observable i, and the sums perform convolutions over N_x points on a grid in x and N_{flav} PDF flavours. Recall that the model parameters are implictly contained in the parton distribution function, detailed by Eq. 4.114. Even in the case of DIS, where the observable is linear in the PDF function, the PDF itself is parametrized as a fully connected neural network and so the full forward map is non-linear in the model parameters. It is conceivable that with the appropriate parametrization of PDF and DIS only observables, the linear model framework in section 4.3.2 could be realised, in which case the posterior distribution in PDF space could be calculated analytically. Investigation into this is deferred for future study.

Similarly process which involve the interaction between two protons, which is clearly relevant for LHC observables, take the form

$$\mathcal{G}\left(u\right)_i = \sum_{\beta}^{N_x} \sum_{l}^{N_{\text{flav}}} A_i^{\alpha\beta lm} f_l(x_\alpha, Q_0) f_m(x_\beta, Q_0|u) \ , \tag{4.116}$$

where now the FK table for observable i is a 4-dimensional tensor $A_i^{\alpha\beta lm}$, and

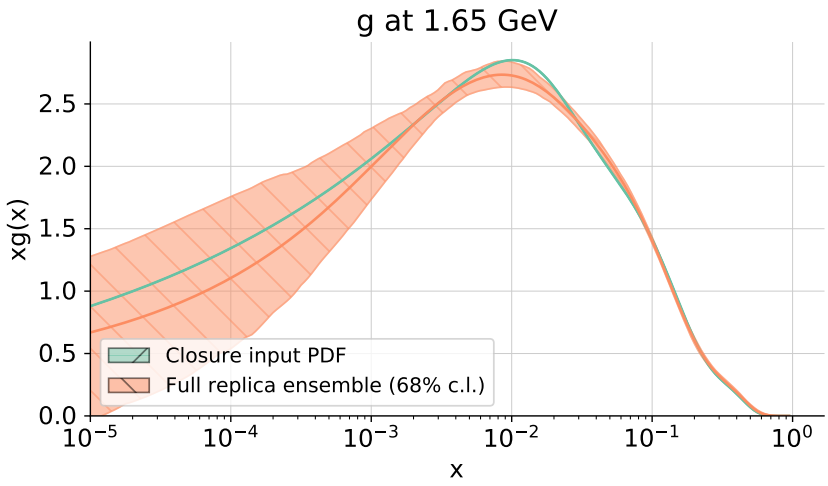

Figure 4.3 *The green line is the input underlying law for the gluon PDF, which is sampled from the ensemble from a fit to data. The 68% confidence interval is plotted for those replicas as the orange band.*

there are sums over x and flavour combinations for both PDFs.

4.5.2 Closure test setup

As input to the closure test, a single replica was drawn randomly from a previous NNPDF fit to experimental data. We refer to this as the underlying law and the corresponding predictions the true observable values. An example of the gluon input is provided in Fig. 4.3. In principle any function could be used as underlying law, however it makes sense to use a realistic input. To make things explicit, the true observables values which are used as input to the closure fit are then given by

$$f = \mathcal{G}(w) \tag{4.117}$$

where $w \in \{u_*\}$, i.e the underlying law parameters are drawn from a pre-existing set of model parameters from performing a fit.

The observables used in the fits are a subset of the full NNPDF4.0 dataset. For convenience, we chose to fit the PDFs on a variant of the NNPDF3.1 dataset used in [159], which is described in detail in a study of the determination of the strange PDF [160]. The datasets used in the calculation of statistical estimators are the new datasets are included in NNPDF4.0. For a full summary of observables used in the test data and a visual representation of the kinematic region of both the

training and testing data, see App. C.2.

The choice of data for both fitting and testing is considered unimportant, one could consider splitting the data into training and test in a way which considered kinematic coverage rather than this naive chronological splitting. Alternatively, since the data is generated from the theory predictions produced by the input underlying law, one could even produce completely artificial data using a different set of FK tables. From a practical standpoint, using the NNPDF3.1 dataset and validating on the newly included datasets in 4.0 allowed us to validate the PDF uncertainties on data outside of the kinematic coverage of data included in the fit. Furthermore, the data estimators only give us local information on the PDF uncertainties and it seems logical to split the data in this way since the results seem more applicable to the reality of how the PDFs end up being used.

We then generate 30 different sets of experimental central values (or L1 data), as discussed in Sec. 4.3.3, for the fitted 3.1-like dataset. Each set of experimental central values was then fitted following NNPDF4.0 methodology [6], producing 40 pseudo-data replicas.

4.6 Results

4.6.1 Bias-variance ratio

We calculated $\mathcal{R}_{bv}$ on the test data, shown in Tab. C.1. An uncertainty on $\mathcal{R}_{bv}$ by performing a bootstrap sample [42], where we randomly sample from both fits and replicas and re-calculate $\mathcal{R}_{bv}$, the value and error presented in the table is then the mean and standard deviation across bootstrap samples. We checked that the distribution of the estimator across bootstrap samples is indeed Gaussian. We also checked that increasing the number of fits and replicas reduced the bootstrap error but the central values were the same within the estimated bootstrap uncertainties.

One can also compare qualitatively the distribution of bias across fits, to the distribution of the difference between replica predictions and expectation values of predictions (in units of the covariance) across different fits and replicas. The square root ratio of the mean of these two distributions is precisely $\mathcal{R}_{bv}$.

	$\mathcal{R}_{bv}$
Total	1.03 ± 0.05

Table 4.1 *The bias-variance ratio, $\mathcal{R}_{bv}$, for unseen data, summarised in Tab. C.1. The uncertainty is estimated by performing a bootstrap sample across fits and replicas and calculating the standard deviation. We see that overall $\mathcal{R}_{bv}$ is consistent with 1, within uncertainties. This gives a good indication that, at least for the unseen data used in this study, the uncertainties are faithful.*

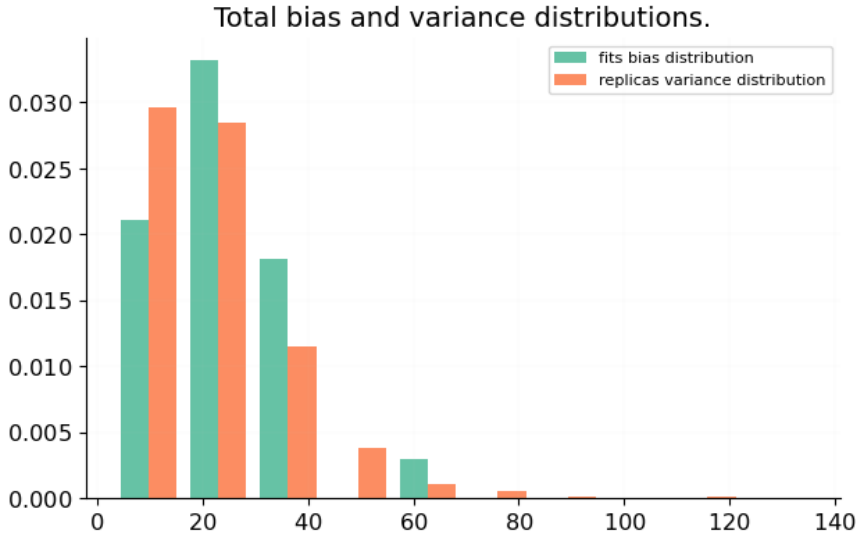

Figure 4.4 *The green histogram is the distribution of the total bias across fits, the orange histogram is the distribution of the difference between the replica and central predictions squared, in units of the covariance across all fits and replicas. This gives a qualitative picture of the full distribution, in Tab. 4.1 we compare the square root of the mean of each distribution.*

4.6.2 Comparison to quantile statistics

As discussed in Sec. 4.4.3, one can define an analogous estimator in data space, based upon $\xi_{n\sigma}$, which was defined on a grid of points in x and Q^2 in PDF space in [2]. There is not a one-to-one correspondence between this and $\mathcal{R}_{bv}$, but a loose approximation using Eq. 4.100. In Tab. 4.2 we compare the estimated $\xi_{1\sigma}$ from substituting $\mathcal{R}_{bv}$ into Eq. 4.100 and to the measured value.

Despite the assumptions entering each of the two estimators differing, we see good agreement between the $\xi_{1\sigma}$ estimated from $\mathcal{R}_{bv}$ and that measured directly. We find this result reassuring, since it indicates not only that the total uncertainty

	$\xi_{1\sigma}$	$\mathrm{erf}(\mathcal{R}_{bv}/\sqrt{2})$
Total	0.69 ± 0.02	0.67 ± 0.03

Table 4.2 *Comparing the measured value of $\xi_1\sigma$ and the estimated value from $\mathcal{R}_{bv}$. The two values are consistent, which suggests the approximation that the ratio of uncertainties is approximately the same across all data is not completely invalidated. Not only are the measured value and estimated value from $\mathcal{R}_{bv}$ self consistent, but they are also consistent with* 0.68*, which further supports the argument that the model uncertainties are faithful.*

averaged across all data is faithful, but also that the uncertainty on each data point seems faithful. If the results differed it would indicate some kind of imbalance, where some components of the uncertainty are correctly represented by the replicas but other directions are not.

4.7 Summary

We've presented a formal framework for inverse problems, from a Bayesian perspective. In particular, the framework provides a more formal description of what it means when we talk about propagating experimental uncertainties into the space of models. Strictly speaking, there is no fitting required to obtain the expression for the posterior distributions in the space of the data or the model, instead these are obtained by marginalising the prior distribution. We note that sampling from the posterior distribution of the model is, in general, highly non-trivial but show that at least for linear problems the NNPDF MC methodology can be shown to produce a sample of models which are distributed according to the posterior model distribution. Furthermore, we provide evidence that even for non-linear models this result at least holds as a good approximation close to the MAP estimator.

We then use this formal framework to think about some of the estimators which we use as part of the NNPDF closure test. In particular we derive bias and variance from decomposing an out of sample error function, which is understood from a classic fitting point of view. The estimators are then related back to the posterior distributions in the Bayesian framework. We note that the estimators themselves are not perfect and suffer from only testing the model uncertainties locally (in regions where the test data probes). Furthermore, the estimators only give an approximate overall picture, and cannot be used to diagnose where the problem arises if we find evidence that the model uncertainties are not faithful.

Given the framework set out here, future work should be undertaken generalise the closure estimators to model space. This would likely involve a combination of the closure estimators presented here and the extension of the Bayesian framework to infinite-dimensional spaces.

We give some preliminary closure results, as a proof of principle, the results presented here used the full NNPDF4.0 fitting methodology [161] and the closure fits were fitted and analysed using observables which will be used as part of the full NNPDF4.0 dataset. The results serve as an example of how the data space estimators can be practically included even in a rather complex setting. Furthermore, since the NNPDF4.0 methodology passes the closure test according to the estimators presented here, we provide evidence that the NNPDF MC approach to inverse problems is a reasonable approach to sampling from the

posterior distribution of the model. In particular, for unseen data the current NNPDF methodology appears to provide faithful uncertainties. The estimators are not limited to this specific application, and the results presented demonstrate how the data space estimators can be incorporated into any inverse problem.

In the closure test framework described in this study, the prior on the data is fully consistent with the generation of the observable central values from the true observables and uncertainties by construction, which is likely not the case in real world fits. Something which should be investigated is guarantee faithful uncertainties when this is no longer the case, for example one could consider a closure test where the generated data was inconsistent with the prior. Most observed inconsistency between the prior and the data central values is likely due to missing theoretical uncertainties, but once all sources of theoretical uncertainties have been accounted for there could still be tension in the data. The advantage of viewing inverse problems from a Bayesian perspective is access to methodologies which deal with inconsistent data (or unknown systematics) in a Bayesian framework, for example in these cosmological studies [162, 163]. These methods potentially offer a more formal approach to dealing with inconsistent data, rather than ad hoc procedures.

In NNPDF fits, the inclusion of MHOUs in the likelihood was justified from a Bayesian perspective [146], but here we have drawn a connection between the model replicas and the posterior distribution in model space, which explains why theoretical uncertainties must be included in both the sampling of the pseudodata replicas and the likelihood. Therefore we emphasise that the framework set out here is not only useful for understanding model uncertainties with the current methodology, but also for motivating future methodological development from a Bayesian perspective.

Chapter 5

Conclusion

This thesis has explored machine learning tools which can be used for sampling various probability distributions which appear in physics.

In the Chapter 2 we trained an Restricted Boltzmann Machine in order to generate binary spin configurations for the Ising model. Here we used our knowledge of the physical model to help us understand the training procedure. In particular, we could see how the estimation of physical observables from configurations generated from the model evolved throughout the training. We can see the training converge as the observables converge to the expected values. We used this to help inform a recipe for training RBMs. We also used annealed importance sampling in order to estimate the normalized likelihood of the training data. Similarly to the observables, we can measure this whilst training the model in order to determine whether the training has converged. We note that the estimation of the likelihood is not restricted to physical systems, and therefore is a more general indication of training convergence than estimating physical observables.

The key result of this work was deriving the physical coupling of the model distribution from the model parameters. In particular we showed that in the case of the RBM trained on states from the 2-D Ising model, the 2-point interaction both reproduced the nearest neighbour interaction structure and the value of the coupling. Being able to extract information from trained ML models is of interest to a wide range of scientific communities. Whilst this result is limited to RBMs, it provides a framework for training models to generate states for any binary system and then extract any N-point interaction that the model learns from the data directly from the model weights. This result gives us a closed form expression

which allows us to understand exactly what the model has learned and therefore, at least for RBMs, is work in the direction of explaining what the ML model is doing instead of treating it as a black box. Being able to derive results such as this for a wider range of models is relevant for a wide range of fields. Since performing this work, there has already been further effort to derive results which are analogous to our N-point interactions derived for the RBM but are *model independent*, an application is shown for simulated individual-level human DNA and traits [164].

In the next Chapter we shifted to a different generative model, the normalizing flow, and showed how we could reproduce the results in [1] with substantially less computational cost. Whilst there were similarities with the techniques used in this work and the work in Chapter 2, the aim of this work was to use the ML model itself as an efficient tool for generating states according to a scalar lattice theory with a ϕ^4 interaction. Our contribution to this line of work was to introduce more flexibile transformations to the flow and perform a more extensive study of the impact of different hyperparameters on the trade-off between the ability of the model to produce samples similar to the target distribution and the time taken to train it. Despite a very large reduction to the amount of effort required to train the normalizing flow compared to the original proof of principle study, we show that the scaling of the training cost is exponential with the lattice size and so whilst this model seems very promising as a supplementary tool for generating lattice field theory configurations, there is clearly more work required in order to understand exactly how the training scales and whether this poor scaling can be overcome. Overcoming this poor scaling of the training should be the focus of future work.

In the final Chapter we presented a formal Bayesian framework for Inverse problems. Investigating this work involved a paradigm shift from the unsupervised learning in the previous Chapters to the NNPDF MC methodology which relied on the supervised learning technique of regression. In the era of precision physics, there is an increased urgency in faithfully representing all uncertainties which includes the uncertainties associate with determining models from finite noisy observations. The Bayesian framework provides a formal description of how to propagate uncertainties from the training data into the model. We show that, at least for linear forward maps, the NNPDF MC methodology of fitting fluctuated data produces a sample of models which are distributed according to the posterior distribution of the model given the data. Further work should be undertaken

to show to what extent this holds for non-linear models. We presented some data space estimators which have an interpretation in classical fitting as different components of the generalisation error. The data estimators can also be used to assess whether the distribution of replicas gives faithful uncertainties in the space of data. Some proof-of-principle results were shown in the context of fitting Parton Distribution Functions to demonstrate the usefulness of both the estimators and the NNPDF MC methodology in generating replicas which appear to have faithful uncertainties on unseen data. Future work should try and connect the closure test estimators and the extension of the Bayesian formalism to the infinite dimensional space of functions. This would potentially give the option of defining estimators which could give a more global understanding of the model uncertainties, since the data estimators are restricted to testing uncertainties in a local sense, where the test data is able to probe.

As a final observation, in Chapter 3, the training involves minimising a reversed Kullbach-Leibler divergence which meant that we were training a model to approximate a known (unnormalized) distribution. Whilst the normalizing flow is a tool which would normally fall under the paradigm of unsupervised learning, this method of training starts to blur the line between the supervised and unsupervised learning. To demonstrate this, consider the Bayesian formalism presented in Chapter 4, where we have an explicit unnormalized expression for the posterior distribution of the model parameters. In that work we focused on the NNPDF MC approach which mixes ideas of bootstrapping and regression to approximately sample from the posterior distribution of the model parameters. But when the two approaches are laid out side by side, we can imagine using a flow model to generate configurations of model parameters from the posterior distribution in model space. The allure of this approach is that we would be able to use the Metropolis algorithm in order to guarantee that the model replicas were indeed sampled from the posterior distribution in model space. This might seem like a rather ambitious goal but recall that with a linear forward map and a Gaussian prior on the data, the posterior model distribution is a multi-variate Gaussian. In this case the flow model has to learn a rotation, rescaling and shifting transformation in order to transform the independent Gaussian latent variables to the multi-variate Gaussian model parameters. We have already demonstrated that the normalizing flows can handle these kinds of transformations, when training the flow model on scalar free theory in Sec. 3.6.1. Showing explicitly that normalizing flows could be used in this way would certainly be an interesting new approach to solving inverse problems and could draw on the formal Bayesian framework

for inverse problems laid out in Chapter 4 and the description training flexible normalizing flow models in Chapter 3.

Machine Learning continues to be a disruptive technology in many fields, and we are sure to continue seeing it be used in the context of Physics. The work in this thesis has allowed us to understand better what distribution an RBM has learned, here we were able to use our understanding of physics to provide a result which is useful for a range of fields which rely on distributions of binary data. We then progressed a line of work which aims to use a similar class of models, normalizing flows, as efficient tools for producing samples from distributions which have a closed form, such as those in lattice field theories. This work massively reduced the computational cost of training these models but also highlighted that there is still a way to go before these kinds of techniques can replace more traditional sampling techniques. In particular the exponential scaling of the training costs must be overcome. Finally we looked at inverse problems from a Bayesian perspective which allowed us to understand better how the NNPDF MC methodology for producing a sample of model replicas relates to the posterior model distribution which emerges as part of this formal framework. Being able to relate statistical estimators used to validate the NNPDF methodology back to this formal framework gives us confidence that our model uncertainties are faithful, and allows us to understand what faithful model uncertainties means.

Appendix A

Appendix: RBM

A.1 The training procedure in more detail: $L^2 = 8 \times 8$

Here we report further details regarding the training of the RBM on the 8×8 Ising model at $T = 1.8$. The specific parameters used for the training are presented in Table 2.3. As mentioned before, the number of constrastive divergence steps k was increased during the training, while the learning rate α was decreased, splitting the training in three phases. The reason for the former prescription, is exemplified in Fig. A.1, where the loss function and the log-likelihood between epoch 3000 and 4000 are plotted for different values of k. While the loss function is left unchanged with varying k, the log-likelihood displays a clear difference in shape, *i.e.* it increases as k is increased, while keeping α fixed at 0.01. If, instead, we choose to keep k fixed at $k = 1$, and decrease α from 0.01 to 0.001, we see no improvement in the log-likelihood as observed on the left hand side of Fig. A.1. Therefore, it is the increase in the value of k that leads to the increase in the log-likelihood, visible in the curves corresponding to $T = 1.8$ of Fig. 2.3.

The reason for reducing the learning rate along the training is visible in Fig. A.2, where we have taken observables for $T = 1.8$ from Fig. 2.4, magnified and normalized by their expected value from Magneto. We therefore observe that decreasing the learning rate implies a reduction of the fluctuations between successive epochs, resulting in a fine tuning of our predictions. This is more evident starting from epoch 3000, where the learning rate is decreased by factor 10.

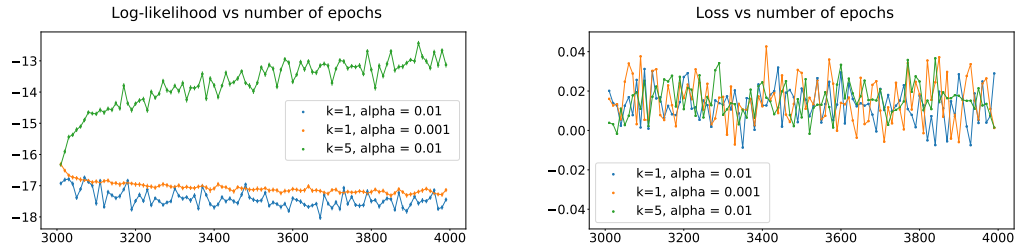

Figure A.1 *From left to right, log-likelihood and loss function between 3000 and 4000 epochs for three different values of k. While the former shows different behaviors, keeping an increasing trend just for the highest k value, the latter doesn't change at all, always remaining near zero.*

As shown in the plots, at the end of the training, the predictions from the RBMs are compatible, within statistics, with the expected value from Magneto. Also, when we approach the end of the training, all the machines shown in the plots are statistically equivalent to each other, since their predictions are all compatible within 2 sigma.

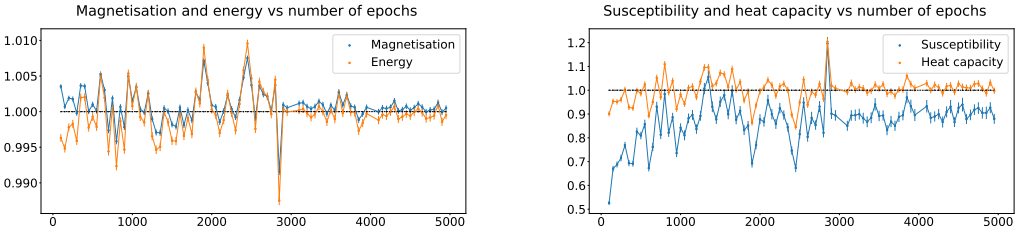

Figure A.2 *Observables for $T = 1.8$ normalized by their expected values as a functions of the training epoch. Magnetization and energy are shown on the left, susceptibility and heat capacity are on the right.*

A.2 Training on a larger system: $L^2 = 16 \times 16$

Here we report the results for a training the RBM on 16×16 Ising configurations, highlighting the main difficulties arising for larger systems. As done in the case of the 8×8 systems, we first tune the value of k in order to have an increasing log-likelihood. It turns out that the log-likelihood behaviour is sensitive to the value of k and α much more than in the case of smaller 8×8 systems. This is shown in Fig. A.3, where the log-likelihood as a function of the training epoch is plotted for different values of k and α. On the left hand side we see that using $\alpha = 0.1$, the log-likelihood is not an increasing function of the epochs, even when the value of k is increased. However, decreasing the value of α to 0.01 and using

$k = 5$ we manage to obtain an increasing behaviour which becomes even more evident when using $k = 10$, as shown on the right hand side. The final setting used

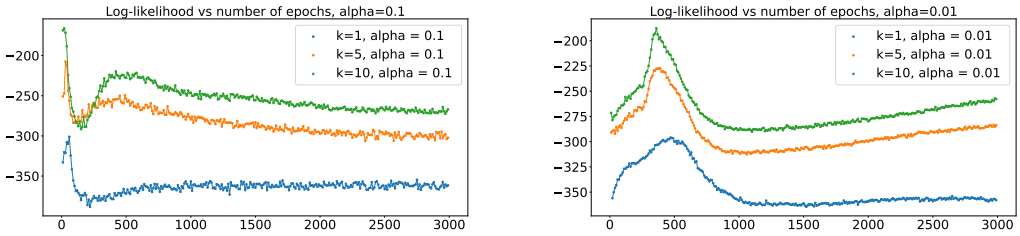

Figure A.3 *Log-likelihood for different values of k and α.*

to train this machine are reported in Table 2.3. The estimators for the training are presented in Fig. A.4 and the observables versus epochs plots in Fig. A.5: all the estimators present the correct behaviour. The initial peak visible in the log-likelihood at about 500 epochs, should not be considered as the point where the machine has been best trained, but rather as an initial fluctuation of the log-likelihood that later on dies out. This observation is also confirmed by the loss function and the reconstruction error, which also present a peak around the same number of epochs, implying that the training has not converged yet at this point. The final increase in the log-likelihood at 8000 epochs, is where we started using $k = 20$ instead of $k = 10$, as reported in Table 2.3.

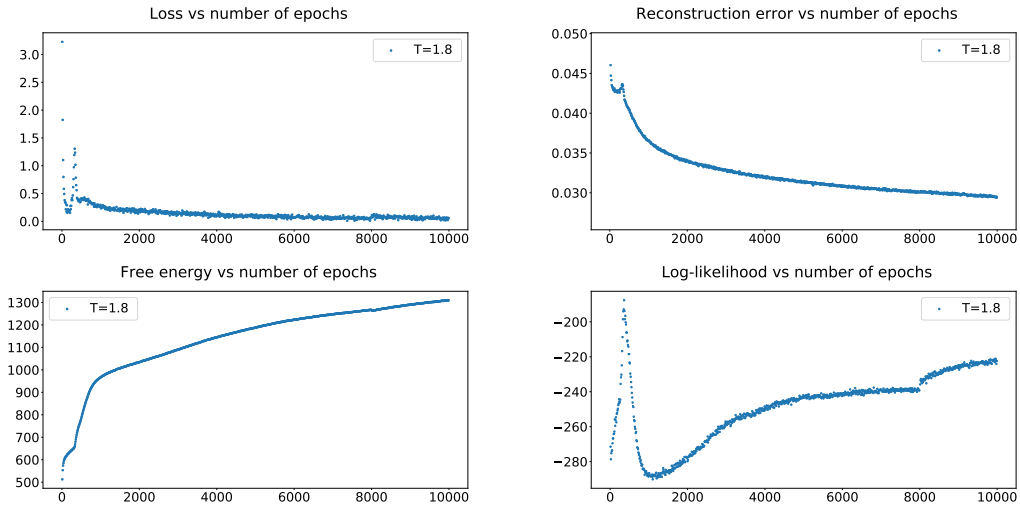

Figure A.4 *Here we observed the increase in the log-likelihood behaviour for our chosen values of k and α, as given in Table 2.3. Both the loss function and reconstruction error decrease as the training progresses.*

From the plot of observables in Fig. A.5, it can be seen that magnetization and heat capacity have almost converged to the expected value, and would only require

a fine tuning by reducing the value of α. The energy and susceptibility are further from the expected values, therefore, longer training is required to obtain more accurate values for these two quantities. Having said that, it appears that all the moments are slowly approaching the expected values, and the fact that the log-likelihood is still increasing with the number of epochs suggests that our machine is indeed learning. We can conclude that for the case of the larger 16×16 system the convergence is much slower than the previous smaller 8×8 systems. We can convince ourselves that the machine is actually learning looking at the

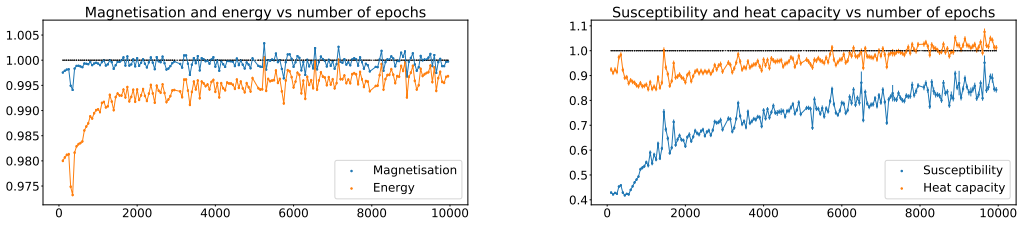

Figure A.5 *Observables vs epochs for $L^2 = 16 \times 16$, $h^2 = 16 \times 16$ and batch size 200. The value of each observable, computed from the RBM, is normalized by its expected value, computed from the training set. Magnetisation (blue) and energy (orange) are plotted on the right hand side, susceptibility (blue) and head capacity (orange) are on the left.*

2-point interaction matrix for the machine at epoch 10000, plotted in Fig.(A.6). Clearly the expected path for the interactions can already be seen in the matrix. The coupling extracted from this matrix is $1/2T \sim 0.258 \pm 0.017$, as compared to the expected value of 0.277. This confirms that the machine agrees with the correct theoretical result within 2 sigma. A more accurate result can be obtained if the machine is trained for a longer period of time, as the log-likelihood is still increasing.

A.3 Changing the batch size

We also examined the effect of changing the batch size for the $L^2 = 16 \times 16$ system, keeping the other hyperparameters the same as the successfully trained machine above, *i.e.* , $\alpha = 0.01$ and $k = 10$. As it can be observed from Fig. A.7, the log-likelihood has a slower increase with batch size 500, as compared to batch size 200.

The slower training in this case can also be observed by plotting the trajectories

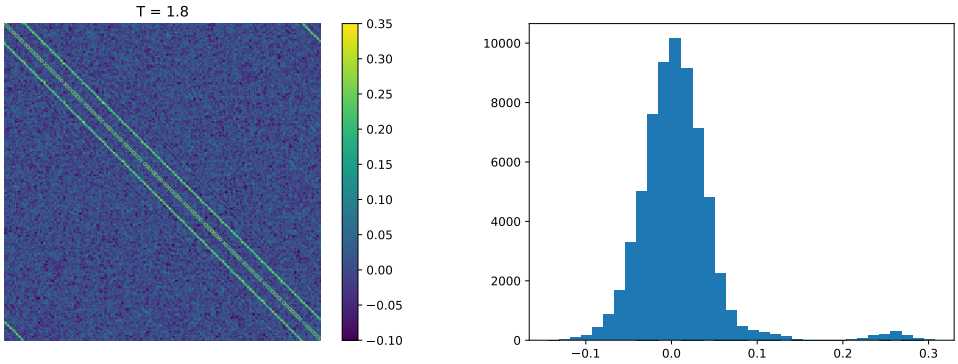

Figure A.6 *The 2-point interaction matrix, H_{j_1,j_2} (left) and its corresponding histogram (right) for the machine with $L^2 = 16 \times 16$, $h^2 = 16 \times 16$ and batch size 200. Again, we observed the larger peak centred around zero, corresponding to non nearest neighbour interactions, while there is a second peak representing the coupling with the nearest neighbour spins.*

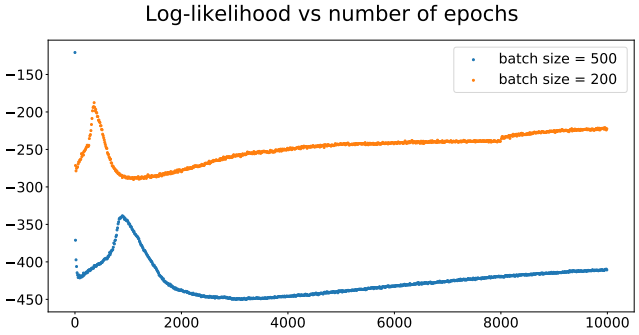

Figure A.7 *The dependence of log likelihood on batch size. The curves correspond to $L^2 = 16 \times 16$ lattice with batch size 500 (blue) and batch size 200 (yellow). The choice of a smaller batch size, results in a steeper rise to the log-likelihood.*

of the observables per epoch, in Fig. A.8, as compared to Fig. A.5. This is also

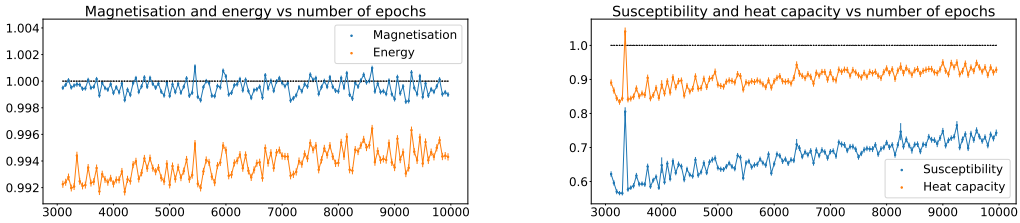

Figure A.8 *Observables, normalized by their expected values, vs epochs for $L^2 = 16 \times 16$, $h^2 = 16 \times 16$ and batch size 500. We observe that it takes the observables longer to converge to the correct values, as compared to the case where a smaller batch size is used, e.g. , see Fig. A.5.*

consistent with the two-point interaction matrix presented in Fig. A.9, which is more noisy in this case as compared to Fig. A.6.

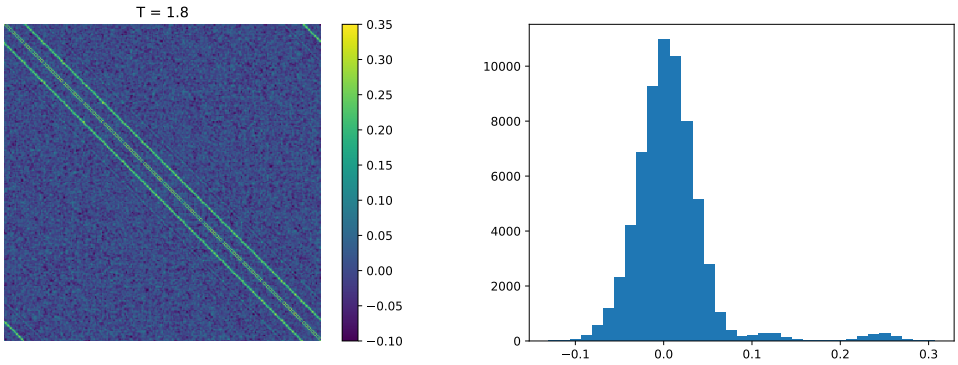

Figure A.9 *The 2-point interaction matrix, H_{j_1,j_2} (left) and its corresponding histogram (right) for the machine with $L^2 = 16 \times 16$, $h^2 = 16 \times 16$ and batch size 500. There is a large peak centred around zero, corresponding to non nearest neighbour interactions, however, a second smaller peak can also be observed next to it. As already discussed, the machine with a larger batch size, i.e. 500, has to be trained for longer epochs as compared to the machine with batch size 200, in order to learn that non nearest neighbour interactions are zero. Finally, the distinct peak on the right hand side of the plot represents the expected coupling with the nearest neighbour spins, compare with Fig.A.6*

A.4 Changing the number of hidden nodes

We also attempted to train an RBM, for the $L^2 = 16 \times 16$ Ising system, where the number of hidden nodes were less than the visible nodes. More explicitly, we chose $h^2 = 12 \times 12$. The log-likelihood is plotted in Fig. A.10. The first 4500 epochs were trained using $\alpha = 0.01$ and $k = 10$, as these settings were successful in training the $h^2 = 16 \times 16$ case in Sec. A.2. However, from the experience gained in Sec. A.3, the batch size was reduced to 50 in order to give an faster increasing slope to the log-likelihood.

The observables, normalized by their expected values, are plotted in Fig. A.11. We can see that the values are furthest from one as compared to the other cases we have trained so far. This is consistent with the measurement of the coupling matrix at a representative epoch towards the end of this training, as presented in Fig. A.12. The nearest neighbour structure is present, however, further correlations

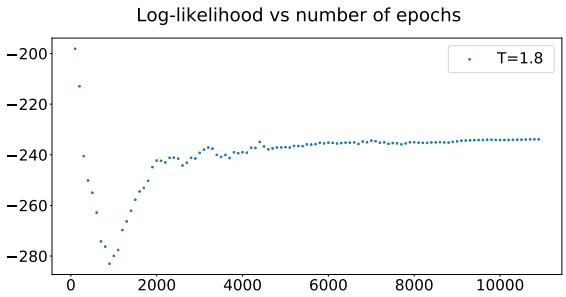

Figure A.10 *Log-likelihood for an RBM with less hidden nodes than visible nodes, $L^2 = 16 \times 16$ and $h^2 = 12 \times 12$. The first 4500 epochs were trained using $\alpha = 0.01$, $k = 10$ and batch size 50. According the prescription, we then reduced the value of α and increased k, i.e. , From 4500 to 8000, we set $\alpha = 0.001$ and $k = 20$. From 8000 to 8700 epochs α and k were kept fixed at their previous value, while the batch size was increase to 200, in order to reduce the fluctuations in the estimate of the log-likelihood. In the last steps we chose $\alpha = 0.0001$ and $k = 30$, and $\alpha = 0.00001$ and $k = 40$.*

between the spins can also be observed, which are not expected. A third peak in the histogram confirms the latter observation more clearly in Fig. A.12.

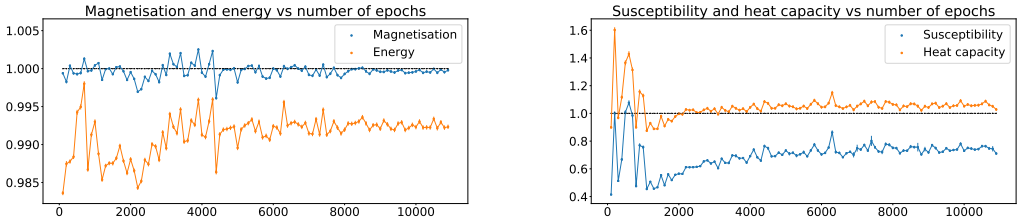

Figure A.11 *Observables vs epochs for $L^2 = 16 \times 16$, $h^2 = 12 \times 12$. It can be observed that the machine has to run for more epochs for it to learn the observables and hence the correct structure.*

According to Fig. A.10, the log-likelihood is still increasing, however, the run has become computationally expensive at this stage with $\alpha = 0.0001$ and $k = 40$. Recall that in Fig. 2.10, it is observed that there are correlations which are further than nearest neighbours predicted by the RBM, before the learning is complete. Therefore we could expect that with more training, the RBM may eventually learn the correct nearest neighbour structure, setting the rest of the correlations to zero. Also notice that a $L^2 = 16 \times 16$ with hidden nodes of the size $h^2 = 12 \times 12$ has $\mathcal{O}(3 \times 10^5)$ less parameters than the case with $h^2 = 16 \times 16$. Hence, the difficulty in training the machine is not perhaps too surprising. Having said that, further works need to be done to gain a better understanding of the effect of reducing

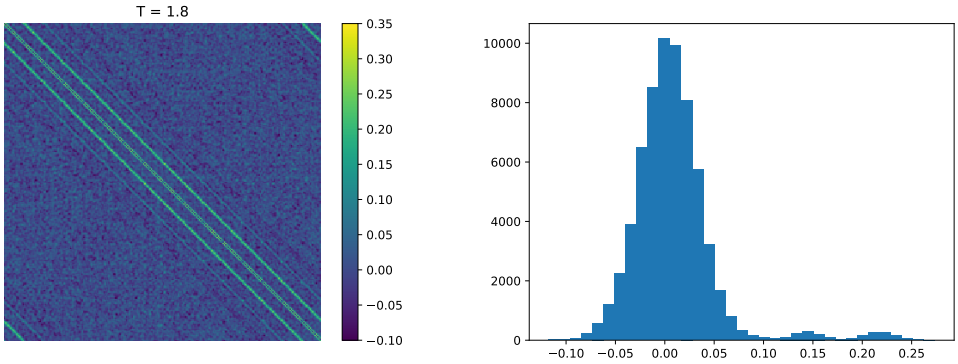

Figure A.12 *The two-point interaction matrix for the $L^2 = 16 \times 16$ system (left) and the corresponding histogram (right) with $h^2 = 12 \times 12$. The first peak is centred around zero, corresponding the the non nearest neighbour interactions. The second peak around 0.15 indicates other non nearest correlations that the machine has to learned to set to zero and are expected to vanish as it trains further. The final peak on the right hand side, corresponds to the correct nearest neighbour coupling.*

hidden nodes on training an RBM.

A.5 3- and 4-point interaction histograms

We present the histograms of the 3- and 4-point couplings between the spins, in Fig. A.13 and A.14 respectively. Generally, the single peak expected behaviour is observed. However for some of the higher temperatures, a second mode seems to appear in the histogram of the three-point interaction. More work needs to be done to understand why these machines appear to be learning a higher order interaction than we would not expect.

A.6 Metropolis history plots

The absolute magnetisation and energy histograms of Ising configurations, over which the final measurements for the observables are made, are presented for two different temperatures in Fig. A.15 and Fig. A.16. The histograms compare well to their corresponding normal distribution, whose parameters are measured by measuring the mean and the standard deviation from the data producing the histogram. Running the Metropolis chain longer than 1×10^6 steps did not change

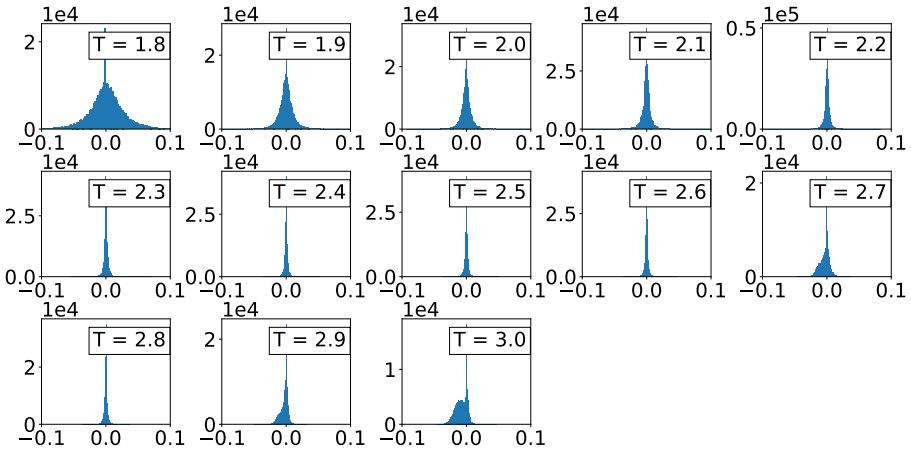

Figure A.13 *The histograms of the entries of the 3-point interaction tensor extracted from RBMs trained at a temperature indicated above each subplot.*

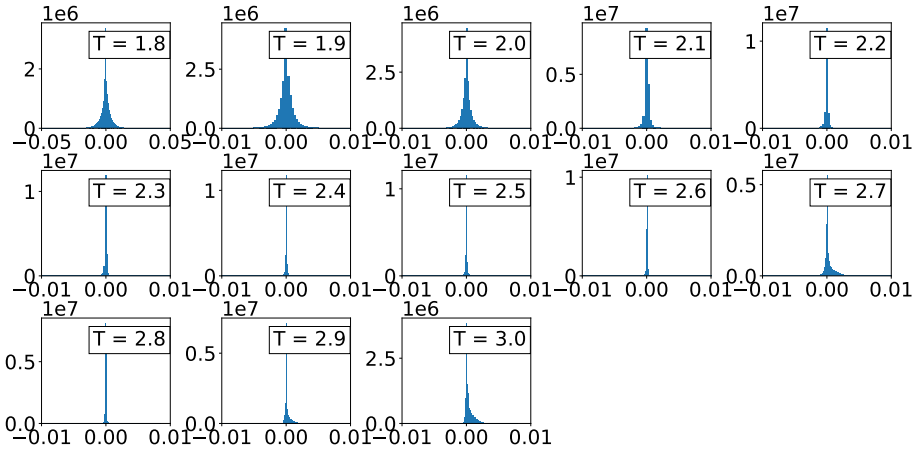

Figure A.14 *The histograms of the entries of the 4-point interaction tensor extracted from RBMs trained at a temperature indicated above each subplot.*

the from of the distributions.

Note that in order for the measurements to be independent of each other, we have binned every 50 measurements of $|m|$ and E on each configuration. To decide on the bin size, we plotted the error on the measurement of magnetisation vs choice of bin size. This is shown in Fig. A.17, while the autocorrelation time, defined in Eq, 2.43, is plotted in Fig. A.18

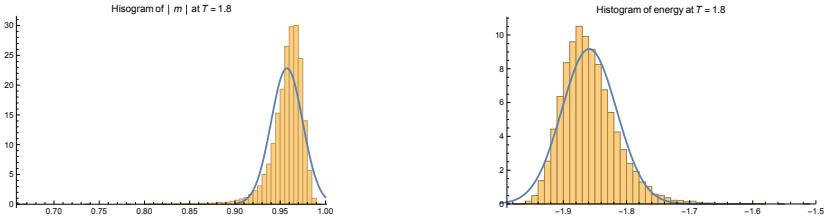

Figure A.15 *Histogram of $|m|$ (left) and energy (right) for the Metropolis algorithm at $T = 1.8$. The blue line represent the normal distribution with values of its mean and standard deviation obtained from the data producing the histogram.*

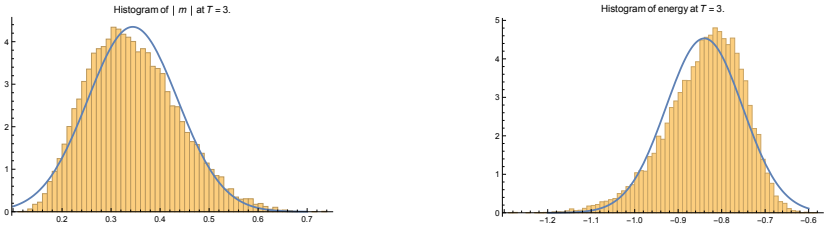

Figure A.16 *Histogram of $|m|$ (left) and energy (right) for the Metropolis algorithm at $T = 3.0$. The blue line represent the normal distribution with values of its mean and standard deviation obtained from the data producing the histogram.*

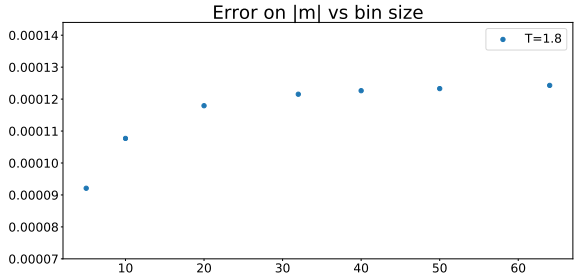

Figure A.17 *Error on $|m|$ plotted for choice of bin size. As the measurements become more independent, the correlation between them decrease and hence the error increases. When the measurements are no longer dependent, the error remains constant.*

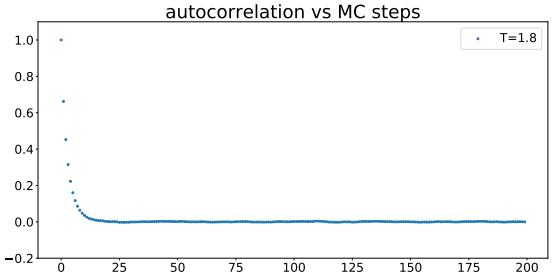

Figure A.18 *Autocorrelation time as a function of MC steps.*

Appendix B

Appendix: Normflow

B.1 ϕ^4 theory on the lattice

The standard ϕ^4 action in two-dimensional Euclidean space is

$$S[\varphi] = \int d^2x \left[\frac{1}{2} \big(\partial_\mu \varphi(x)\big) \big(\partial_\mu \varphi(x)\big) + \frac{1}{2} m_0^2 \varphi(x)^2 + \frac{1}{4!} g_0 \varphi(x)^4 \right], \tag{B.1}$$

where m_0 is the bare mass and g_0 is the bare coupling for the quartic interaction term.

We can define a discretised analogue of this theory on a periodic lattice Λ with lattice spacing a and spatial extent $L_\mu = aN_\mu$ using the following steps:

1. Use the vanishing boundary term (due to periodicity) to replace the derivative term with the Laplacian,

$$\big(\partial_\mu \varphi(x)\big) \big(\partial_\mu \varphi(x)\big) \to -\varphi(x) \partial^2 \varphi(x)\,. \tag{B.2}$$

2. Adopt the following discretised version of the Laplacian:

$$\begin{aligned}\partial^2\varphi_x &\to \delta^2\varphi_x \\ &= \frac{1}{a^2}\sum_{\mu=1}^{2}(\varphi_{x+ae_\mu}+\varphi_{x-ae_\mu}-2\varphi_x)\,,\end{aligned} \tag{B.3}$$

   where e_μ represents a unit vector in the μ-th dimension.

3. Replace the integral with a sum,

$$\int d^2x \to a^2\sum_{x\in\Lambda}\,. \tag{B.4}$$

4. For convenience, define dimensionless couplings $m_0^2 \to m_0^2a^2$ and $g_0 \to g_0a^2$.

This leads to the following lattice action:

$$S(\varphi)=\sum_{x\in\Lambda}\left[\frac{1}{2}\varphi_x(-\delta^2+m_0^2)\varphi_x+\frac{g_0}{4!}\varphi_x^4\right]. \tag{B.5}$$

Using (B.3) and the translational invariance of the action yields

$$S(\varphi)=\sum_{x\in\Lambda}\left[-\sum_{\mu=1}^{2}\varphi_x\varphi_{x+e_\mu}+(2+\frac{m_0^2}{2})\varphi_x^2+\frac{g_0}{4!}\varphi_x^4\right]. \tag{B.6}$$

Equation (3.39) is related to (B.6) through

$$\varphi=\sqrt{\beta}\phi,\qquad 2+\frac{m_0^2}{2}=\frac{1-2\lambda}{\beta},\qquad \frac{g_0}{4!}=\frac{\lambda}{\beta^2}\,. \tag{B.7}$$

B.2 Estimation of integrated autocorrelation time

In practice, the integrated autocorrelation time defined by Equation (3.4) must be estimated from a Markov chain of finite length N. However, the statistical error on the autocovariance estimator,

$$\hat{\Gamma}_{\mathcal{O}}(t)=\frac{N}{N-1}\left[\frac{1}{N}\sum_{n=1}^{N-t}\mathcal{O}(\phi^{(n+t)})\mathcal{O}(\phi^{(n)})-\bar{\mathcal{O}}^2\right], \tag{B.8}$$

increases with t, so it is preferable to truncate the sum at some separation $W < N$.

We thus have the estimator

$$\hat{\tau}_{\mathrm{int},\mathcal{O}}(W) = \frac{1}{2} + \sum_{t=1}^{W} \frac{\hat{\Gamma}(t)}{\hat{\Gamma}(0)}\,, \tag{B.9}$$

and must attempt to find the value of W which minimises the sum of:

1. The bias due to truncating the sum,

$$\begin{aligned}\varepsilon_{\mathrm{trunc}}(W) &\equiv \mathrm{bias}\big[\hat{\tau}_{\mathrm{int},\mathcal{O}}(W)\big] \\ &= -\sum_{t=W+1}^{\infty} \frac{\Gamma(t)}{\Gamma(0)} \approx -\tau_{\mathrm{int},\mathcal{O}} e^{-W/\tau_{\mathcal{O}}}\,,\end{aligned} \tag{B.10}$$

   where we have assumed that W is sufficiently large that the autocorrelation takes a pure exponential form, with $\tau_{\mathcal{O}}$ being the characteristic relaxation time of the slowest mode of $\mathcal{O}$.

2. The statistical error approximated by the Madras-Sokal formula [63],

$$\varepsilon^2_{\mathrm{stat}}(W) \equiv \mathrm{var}\big[\hat{\tau}_{\mathrm{int},\mathcal{O}}(W)\big] \approx \frac{2(2W+1)}{N}\tau^2_{\mathrm{int},\mathcal{O}}\,, \tag{B.11}$$

   which uses the approximation $\tau_{\mathcal{O}} \ll W \ll N$.

We would therefore like to find the minimum of $\Delta(W) = |\varepsilon_{\mathrm{trunc}}(W)| + |\varepsilon_{\mathrm{stat}}(W)|$. To do so we follow the 'automatic windowing' procedure detailed in Reference [165] (Sec. 3.3).

First, note that we can re-cast the integrated autocorrelation time in terms of the equivalent pure exponential decay,

$$\tau^*_{\mathrm{int},\mathcal{O}} = -\left[\log\left(\frac{2\tau_{\mathrm{int},\mathcal{O}} - 1}{2\tau_{\mathrm{int},\mathcal{O}} + 1}\right)\right]^{-1}, \tag{B.12}$$

which is equal to zero, rather than 1/2, for uncorrelated data, and so offers improved precision in situations where decorrelation occurs very quickly.

We furthermore assume that it is valid to substitute the slowest mode $\tau_{\mathcal{O}}$ for $\lambda\tau^*_{\mathrm{int},\mathcal{O}}$ with λ being a small constant factor, to be tuned such that smallest value of W for which

$$\lambda\frac{\partial\hat{\Delta}(W)}{\partial W} = -e^{-W/(\lambda\hat{\tau}^*_{\mathrm{int},\mathcal{O}})} + \frac{\lambda\hat{\tau}^*_{\mathrm{int},\mathcal{O}}}{\sqrt{WN}} \tag{B.13}$$

drops below zero occurs, generally, at a point at which $\hat{\tau}_{\text{int},\mathcal{O}}(W)$ levels off to a plateau.

Since we generally encountered very small integrated autocorrelation times, the approximation in Equation (B.10) is probably a poor one. By the same token, however, statistical errors were minimal. Ultimately, a little tuning of λ by visual inspection of $\hat{\tau}_{\text{int},\mathcal{O}}(W)$ for a small number of experiments was sufficient.

B.3 Comparison with literature results

Albergo et al. [1] used flows comprising solely affine coupling layers parametrized by fully-connected networks to generate ϕ^4 configurations on lattices ranging from 6^2 to 14^2 sites. They reported training times of 1-2 GPU-weeks in order to reach an average acceptance rate of 70% in the Metropolis-Hastings phase.

Although theirs was a proof-of-principle study, we were nonetheless curious to check how quickly our hybrid affine/spline models could be trained to reach a 70% acceptance rate, using the same action and couplings provided in Reference [1] (though our previous results for $|\Lambda| = 6^2 - 14^2$, $\lambda = 1/2$ and $L/\xi = 4$ correspond to essentially the same systems). We trained these models on a desktop PC. The real time taken to train models to reach at least 70% acceptance are given in Table B.1.

Presumably, the main reason that the times reported in the original study are so much larger than than what we find is the use of much larger neural networks; for the 14^2 lattice they used networks with width 1024 and depth 6 whereas for the same system our models used comparatively minuscule networks of width 196 and depth 2 (i.e. a single hidden layer). Still, given the near order-of-magnitude increase in real train times for each lattice size increase, with this setup and a target of 70% acceptance we would likely encounter times measured in weeks as soon as $L = 18$ or 20. Despite the modest hardware, it is abundantly clear that scaling this technique up to system sizes at which critical slowing down becomes a serious problem will require significantly more than fine-tuning the current approach.

L	H	N_{affine}	N_{segments}	N_{batch}	N_{epoch}	η_0	time
6	36	2	8	300	350	0.02	9s
8	64	2	8	500	750	0.01	37s
10	100	3	8	1250	2000	0.005	5.0m
12	144	3	8	4000	5000	0.003	43m
14	196	3	8	7000	14000	0.001	4.5h

Table B.1 *Measurements of the real time taken to train our models to reach an acceptance rate of 70% for the systems studied in Reference [1]. L is the lattice length, H is the number of hidden nodes in the neural networks, N_{affine} is the number of affine layers, N_{segments} is the number of segments used in the single spline layer, N_{batch} is the number of states in a batch, N_{epoch} is the number of training iterations. Since we were aiming for speed of training rather than reaching the highest possible acceptance rates, we increased the initial learning rate η_0 with respect to our main study, although we do not recommend doing this in general. The models were trained on a desktop PC with an Intel i7-7700K quad-core CPU and 16GB RAM.*

Appendix C

Appendix: Inverse Problems

C.1 Gaussian integrals

Theory errors can be included in this framework by allowing the distribution of observables around the theory prediction to have a finite width, *e.g.* by replacing the Dirac delta

$$\delta(y - \mathcal{G}u) \tag{C.1}$$

in Eq. 4.12 with a Gaussian

$$\theta(y, u|\mathcal{G}) \propto \exp\left[-\frac{1}{2}\left(y - \mathcal{G}u\right)^T C_T^{-1}\left(y - \mathcal{G}u\right)\right] . \tag{C.2}$$

For the purposes of this study, we do not want to provide a realistic estimate of theory errors. Instead we will be assuming that the errors are uncorrelated and identical for all data points

$$C_T = \sigma^2 \mathbb{1} , \tag{C.3}$$

and we will be interested in the limit where $\sigma^2 \to 0$.

C.1.1 Integrating out the data

Marginalizing with respect to y in this case yields

$$\pi_M(u|y_0, u_0, \mathcal{G}) \propto \pi_M^0(u|u_0) \int dy\, \pi_D^0(y|y_0)\theta(y, u|\mathcal{G}) . \tag{C.4}$$

The argument of the exponential in the integrand is a quadratic form in y,

$$A = (y - y_0)^T C_D^{-1} (y - y_0) + (y - \mathcal{G}u)^T C_T^{-1} (y - \mathcal{G}u) \ . \tag{C.5}$$

The integral can be easily evaluated by completing the square,

$$A = (y - \tilde{y})^T \tilde{C}_D^{-1} (y - \tilde{y}) + R_D \, . \tag{C.6}$$

Comparing Eqs. C.5 and C.6 at order y^2 and y, yields

$$\tilde{C}_D^{-1} = \frac{1}{\sigma^2} \left(\mathbb{1} + \sigma^2 C_D^{-1} \right) \ , \tag{C.7}$$

$$\tilde{y} = \left(\mathbb{1} + \sigma^2 C_D^{-1} \right)^{-1} \left(\mathcal{G}u + \sigma^2 C_D^{-1} y_0 \right) \ , \tag{C.8}$$

and therefore

$$\begin{aligned} \tilde{y}^T \tilde{C}_D^{-1} \tilde{y} = \frac{1}{\sigma^2} (\mathcal{G}u)^T \left(\mathbb{1} + \sigma^2 C_D^{-1} \right)^{-1} (\mathcal{G}u) + y_0^T C_D^{-1} \left(\mathbb{1} + \sigma^2 C_D^{-1} \right)^{-1} (\mathcal{G}u) + \\ + (\mathcal{G}u)^T C_D^{-1} \left(\mathbb{1} + \sigma^2 C_D^{-1} \right)^{-1} y_0 + \sigma^2 y_0^T C_D^{-1} \left(\mathbb{1} + \sigma^2 C_D^{-1} \right)^{-1} C_D^{-1} y_0 \, . \end{aligned} \tag{C.9}$$

Note that the four terms in the equation above are ordered in increasing powers of σ^2 and ultimately we will be interested in the limit $\sigma^2 \to 0$, which reproduces the Dirac delta in $\theta(y, u)$. Plugging Eq. C.9 in Eq. C.6 and again comparing to Eq. C.5, we find

$$\begin{aligned} R_D = \frac{1}{\sigma^2} (\mathcal{G}u)^T \left[\mathbb{1} - \frac{1}{\mathbb{1} + \sigma^2 C_D^{-1}} \right] (\mathcal{G}u) - y_0^T C_D^{-1} (\mathcal{G}u) - (\mathcal{G}u)^T C_D^{-1} y_0 + \\ + y_0^T C_D^{-1} y_0 + \mathcal{O}(\sigma^2) \, , \end{aligned} \tag{C.10}$$

Expanding for small values of σ^2 the terms of order $1/\sigma^2$ cancel; keeping only finite terms in the limit $\sigma^2 \to 0$ we finally obtain

$$R_D = (\mathcal{G}u - y_0)^T C_D^{-1} (\mathcal{G}u - y_0) \ . \tag{C.11}$$

This is exactly the result that we obtained earlier when

$$\theta(y, u|\mathcal{G}) = \delta(y - \mathcal{G}u) \, . \tag{C.12}$$

It should not come as a surprise since in the limit where $\sigma^2 \to 0$ the Gaussian distribution that we chose to describe the fluctutations of the data around the

theory predictions reduces indeed to a Dirac delta. The posterior for the model is exactly the one we computed in Sect. 4.2. We do not learn anything new from this exercise, but it is a useful warm-up for the next example. The integral over y can now be performed easily, since it is yet again a Gaussian integral.

C.1.2 Integrating out the model

Using the same approach as above, we now want to marginalise with respect to the model in order to obtain the posterior distribution of the data:

$$\pi_D(y|y_0, u_0, \mathcal{G}) \propto \pi_D^0(y|y_0) \int du\, \pi_M^0(u|u_0)\theta(y, u|\mathcal{G})\,. \tag{C.13}$$

We follow exactly the same procedure outlined above, starting from the argument of the exponential

$$A = (u - u_0)^T C_M^{-1} (u - u_0) + (y - \mathcal{G}u)^T C_T^{-1} (y - \mathcal{G}u)\ , \tag{C.14}$$

we complete the square and rewrite it in the form

$$A = (u - \tilde{u})^T \tilde{C}_M^{-1} (u - \tilde{u}) + R_M\,. \tag{C.15}$$

It can be readily checked that in this case

$$\tilde{C}_M^{-1} = \frac{1}{\sigma^2}\left(\mathcal{G}^T\mathcal{G} + \sigma^2 C_D^{-1}\right)\ , \tag{C.16}$$

$$\tilde{u} = \left(\mathcal{G}^T\mathcal{G} + \sigma^2 C_M^{-1}\right)\left(\mathcal{G}^T y + \sigma^2 C_M^{-1} u_0\right)\ . \tag{C.17}$$

In order to evaluate R_M, we need

$$\begin{aligned}\tilde{u}^T\tilde{C}_M^{-1}\tilde{u} = \frac{1}{\sigma^2} y^T\mathcal{G}\left(\mathcal{G}^T\mathcal{G} + \sigma^2 C_M^{-1}\right)^{-1}\mathcal{G}^T y + u_0^T C_M^{-1}\left(\mathcal{G}^T\mathcal{G} + \sigma^2 C_M^{-1}\right)^{-1}\mathcal{G}^T y + \\ + y^T\mathcal{G}\left(\mathcal{G}^T\mathcal{G} + \sigma^2 C_M^{-1}\right)^{-1} C_M^{-1} u_0 + \mathcal{O}(\sigma^2)\,.\end{aligned} \tag{C.18}$$

Noting that

$$\left(\mathcal{G}^T\mathcal{G} + \sigma^2 C_M^{-1}\right)^{-1} = \frac{1}{\sigma^2} C_M - \frac{1}{\sigma^2} C_M \mathcal{G}^T \left(\mathcal{G}\frac{1}{\sigma^2} C_M \mathcal{G}^T + \mathbb{1}\right)^{-1} \mathcal{G}\frac{1}{\sigma^2} C_M\,, \tag{C.19}$$

we have, in the limit where $\sigma^2 \to 0$

$$\mathcal{G}\left(\mathcal{G}^T\mathcal{G} + \sigma^2 C_M^{-1}\right)^{-1}\mathcal{G}^T = \mathbb{1} - \sigma^2\left(\mathcal{G}C_M\mathcal{G}^T\right)^{-1} + \mathcal{O}(\sigma^4)\,. \tag{C.20}$$

Collecting all terms we find

$$R_M = (y - \mathcal{G}u_0)^T\left(\mathcal{G}C_M\mathcal{G}^T\right)^{-1}(y - \mathcal{G}u_0)\ . \tag{C.21}$$

Performing the Gaussian integral over u in Eq. C.13, we obtain the posterior distribution of the data

$$\pi_D^y(y) \propto \exp\left[-\frac{1}{2}(y - y_0)^T C_D^{-1}(y - y_0) - \frac{1}{2}(y - \mathcal{G}u_0)^T\left(\mathcal{G}C_M\mathcal{G}^T\right)^{-1}(y - \mathcal{G}u_0)\right]\ . \tag{C.22}$$

As in the case above, we note that this is a Gaussian distribution,

$$\pi_D^y(y) \propto \exp\left[-\frac{1}{2}(y - \tilde{y})^T \tilde{C}_D^{-1}(y - \tilde{y})\right]\ , \tag{C.23}$$

where the mean and the covariance are given in Eqs. 4.50 and 4.51. We can rewrite those expressions as

$$\tilde{y} = \mathcal{G}\tilde{u}\,, \tag{C.24}$$

$$\tilde{C}_D^{-1} = C_D^{-1} + \left(\mathcal{G}C_M\mathcal{G}^T\right)^{-1}\ . \tag{C.25}$$

In order to simplify the notation we introduce

$$\hat{C}_M = \left(\mathcal{G}C_M\mathcal{G}^T\right)\ , \tag{C.26}$$

and then

$$\tilde{C}_D = \hat{C}_M - \hat{C}_M\left(\hat{C}_M + C_D\right)^{-1}\hat{C}_M\,. \tag{C.27}$$

C.2 Closure test setup details

C.2.1 Data

The full list of datasets included in the test set are shown in Tab. C.1. The central values are not actually used in the closure test, however we use the experimental uncertainties in the calculation of both $\mathcal{R}_{bv}$ and $\xi^{(\text{data})}_{1\sigma'}$. The corresponding predictions generated from the underlying law are used as the true observable values. Neither of the data-space closure estimators rely on the central values of the test datasets.

Data set	Ref.
DY E906 $\sigma^d_{\text{DY}}/\sigma^p_{\text{DY}}$ (SeaQuest)	[166]
ATLAS W, Z 7 TeV ($\mathcal{L} = 4.6$ fb^{-1})	[167]
ATLAS DY 2D 8 TeV	[168]
ATLAS high-mass DY 2D 8 TeV	[169]
ATLAS $\sigma_{W,Z}$ 13 TeV	[170]
ATLAS W^++jet 8 TeV	[171]
ATLAS σ^{tot}_{tt} 13 TeV ($\mathcal{L} = 139\,$fb^{-1})	[172]
ATLAS $t\bar{t}$ lepton+jets 8 TeV	[173]
ATLAS $t\bar{t}$ dilepton 8 TeV	[174]
ATLAS single-inclusive jets 8 TeV, R=0.6	[175]
ATLAS dijets 7 TeV, R=0.6	[176]
ATLAS direct photon production 13 TeV	[177]
ATLAS single top R_t 7, 8, 13 TeV	[178–180]
CMS dijets 7 TeV	[181]
CMS 3D dijets 8 TeV	[182]
CMS σ^{tot}_{tt} 5 TeV	[183]
CMS $t\bar{t}$ 2D dilepton 8 TeV	[184]
CMS $t\bar{t}$ lepton+jet 13 TeV	[185]
CMS $t\bar{t}$ dilepton 13 TeV	[186]
CMS single top $\sigma_t + \sigma_{\bar{t}}$ 7 TeV	[187]
CMS single top R_t 8, 13 TeV	[188, 189]
LHCb $Z \to \mu\mu, ee$ 13 TeV	[190]

Table C.1 *Observables included in the test data. We wish to stress that the observable central values themselves are not used, however the experimental uncertainties are used in the definition of the closure estimators, and the corresponding predictions from either the underlying law or the closure fits.*

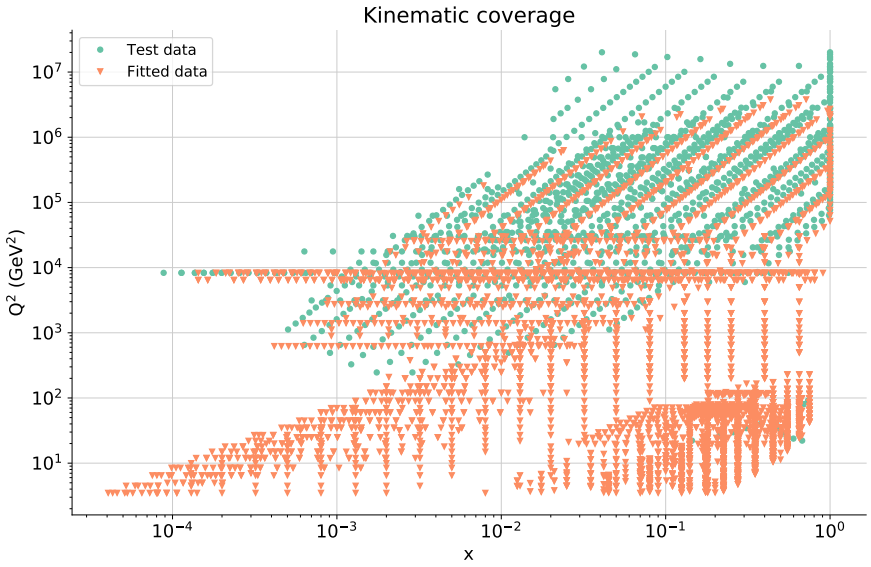

Figure C.1 *The kinematic coverage of the training and test data used to train the models and produce results presented in this paper. We emphasise that the split of datasets was largely chosen on practical grounds, not because of a deep reason to split the data chronologically. The kinematics of the two sets of data with this particular split overlaps but there are also kinematic regions which the test dataset probes, for which there was no training data.*

For completeness, in Fig. C.1 the kinematic coverage of the training datasets, which as mentioned is the NNPDF3.1-like dataset used in [160], and the test datasets shown in Tab. C.1 is plotted.

C.2.2 Models

A summary of the hyperparameters for the neural networks used during the closure fits is given in table Tab. C.2. These hyperparameters were chosen as part of an extensive hyperparameters scan, which will be explained in detail in NNPDF4.0, for this study we simply provide the values of the hyperparameters as a point of reference.

hyperparameter	value
architecture	2-25-20-8
activation	tanh-tanh-linear
minimiser	NAdam
max training length	17000 epochs
pre-processing exponents	not fitted

Table C.2 *Hyperparameters for neural networks used in this study. The parameter choices, and how these choices were made will be discussed in the full NNPDF4.0 paper. The table here is simply to add context to the results below. There are 763 trainable parameters.*

C.3 Understanding NNPDF3.0 data estimators

In the closure test presented in NNPDF3.0 [2] there was a data-space estimator which aimed to measure the level of over or under fitting, Δ_{χ^2}. Here we discuss how Δ_{χ^2} can emerge from the bias-variance decomposition and then use the linear model to try and understand it in the context of viewing the ensemble of model replicas as a sample from the posterior distribution of the model given the data.

Despite the link between the estimators emerging from the decomposition of $\mathcal{E}^{\text{out}}$ and the posterior distribution for data which is not used to inform the model parameters, if we perform the same decomposition as in Sec. 4.4.1 but set ${y_0}' = y_0$ then we find that the cross term in the final line of Eq. 4.84 does not go to zero when the expectation across data is taken because there is a dependence on y_0 in both the model predictions and the noisey data. As a result we have to modify Eq. 4.86 to be

$$\mathbf{E}_{y_0}[\mathcal{E}^{\text{in}}] = \mathbf{E}_{y_0}[\text{bias}] + \mathbf{E}_{y_0}[\text{variance}] + \mathbf{E}_{y_0}[\text{noise}] + \mathbf{E}_{y_0}[\text{noise cross term}]\,, \quad \text{(C.28)}$$

where we refer now to the right hand side of Eq. C.28 as $\mathcal{E}^{\text{in}}$ because it's evaluated on the data used to inform the model replicas.

Now if we examine the definition of Δ_{χ^2} introduced in [2], defined as the difference between the χ^2 between the expectation value of the model predictions and the level one data, and the χ^2 between the underlying observable values and the level one data. In [2] the denominator was also set to be the second term in the numerator, however here we slightly re-define Δ_{χ^2} to instead simply be normalized by the number of data points:

$$\begin{aligned}
&\Delta_{\chi^2} = \\
&\frac{1}{N_{\text{data}}}\Big[\left(\mathbf{E}_{\{u_*\}}\left[\mathcal{G}\left(u_*^{(k)}\right)\right] - y_0\right)^T C_D^{-1} \left(\mathbf{E}_{\{u_*\}}\left[\mathcal{G}\left(u_*^{(k)}\right)\right] - y_0\right) \\
&\qquad - \left(f - y_0\right)^T C_D^{-1} \left(f - y_0\right) \Big] \\
&\qquad = \text{bias} + \text{noise cross term}\,,
\end{aligned} \quad \text{(C.29)}$$

where in the second line we show how Δ_{χ^2} itself can be decomposed to be equal to two of the terms in Eq. C.28.

Constant values of Δ_{χ^2} define elliptical contours in data space centered on the level one data. $\Delta_{\chi^2} = 0$, in particular, defines a contour which is centered on

the level one data and passes through the underlying law. When viewing Δ_{χ^2} from a classical fitting perspective, if $\Delta_{\chi^2} < 0$ then the expectation value of the model predictions fit the level one data better than the underlying observables - which indicates an overfitting of the shift, $\boldsymbol{\eta}$. Similarly, $\Delta_{\chi^2} > 0$ indicates some underfitting of the level one data.

If we return to the linear model we can write the analytic value of Δ_{χ^2}. Firstly, since $y_0{}' = y_0$ we can simplify Eq. 4.105

$$\begin{aligned}\mathbf{E}_{y_0}[\text{bias}] &= \frac{1}{N_{\text{data}}}\text{Tr}\left[\mathcal{G}\tilde{C}_M\mathcal{G}^T C_D^{-1}\right] \\ &= \frac{1}{N_{\text{data}}}\text{Tr}\left[\tilde{C}_M\tilde{C}_M^{-1}\right] \\ &= \frac{N_{\text{model}}}{N_{\text{data}}}\,,\end{aligned} \tag{C.30}$$

because $\tilde{C}_M\tilde{C}_M^{-1}$ is an $N_{\text{model}} \times N_{\text{model}}$ identity matrix. Similarly we can write down the cross term

$$\begin{aligned}\mathbf{E}_{y_0}[\text{noise cross term}] &= \frac{-2}{N_{\text{data}}}\mathbf{E}_{y_0}\left[(\mathcal{G}\tilde{C}_M\mathcal{G}^T C_D^{-1}\eta)^T C_D^{-1}\eta\right] \\ &= \frac{-2}{N_{\text{data}}}\text{Tr}[\mathcal{G}\tilde{C}_M\mathcal{G}^T C_D^{-1}] \\ &= -2\frac{N_{\text{model}}}{N_{\text{data}}}\end{aligned} \tag{C.31}$$

which leaves us with

$$\mathbf{E}_{y_0}[\Delta_{\chi^2}] = -\frac{N_{\text{model}}}{N_{\text{data}}}\,. \tag{C.32}$$

The point is that the linear model has already been shown to be a sample from posterior distribution of the model given the data. But from the classical fitting point of view we would say this model has overfitted.

As such, we do not report any results with Δ_{χ^2} here, because when $\mathcal{R}_{bv} = 1$, it doesn't add much to the discussion. It may still be useful as a diagnostic tool when $\mathcal{R}_{bv} \neq 1$, which as discussed could be for a variety of reasons - including fitting inefficiency. It also may be used as a performance indicator for deciding between two fitting methodologies: if both fits are shown to have $\mathcal{R}_{bv} = 1$, the methodology with smaller magnitude of Δ_{χ^2} could be preferential. The same could be said for bias and variance however, bias in particular is clearly closely related to Δ_{χ^2}.

Bibliography

[1] M. S. Albergo, G. Kanwar, and P. E. Shanahan. Flow-based generative models for Markov chain Monte Carlo in lattice field theory. *Phys. Rev. D*, 100:034515, August 2019.

[2] Richard D. Ball et al. Parton distributions for the LHC Run II. *JHEP*, 04:040, 2015.

[3] Guido Cossu, Luigi Del Debbio, Tommaso Giani, Ava Khamseh, and Michael Wilson. Machine learning determination of dynamical parameters: The ising model case. *Physical Review B*, 100(6), Aug 2019.

[4] Luigi Del Debbio, Joe Marsh Rossney, and Michael Wilson. Efficient modelling of trivializing maps for lattice ϕ^4 theory using normalizing flows: A first look at scalability, 2021.

[5] Luigi Del Debbio, Tommaso Giani, and Michael Wilson. Bayesian Approach to Inverse Problems: an Application to NNPDF Closure Testing. in preparation.

[6] Richard D. Ball, Stefano Carrazza, Juan Cruz-Martinez, Luigi Del Debbio, Stefano Forte, Tommaso Giani, Shayan Iranipour, Zahari Kassabov, Jose I. Latorre, Emanuele R. Nocera, Rosalyn L. Pearson, Juan Rojo, Roy Stegeman, Christopher Schwan, Maria Ubiali, Cameron Voisey, and Michael Wilson. The path to proton structure at one-percent accuracy, 2021.

[7] Richard D. Ball, Stefano Carrazza, Juan Cruz-Martinez, Luigi Del Debbio, Stefano Forte, Tommaso Giani, Shayan Iranipour, Zahari Kassabov, Jose I. Latorre, Emanuele R. Nocera, Rosalyn L. Pearson, Juan Rojo, Roy Stegeman, Christopher Schwan, Maria Ubiali, Cameron Voisey, and Michael Wilson. An open-source machine learning framework for global analyses of parton distributions, 2021.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[10] Joshua Susskind, Adam Anderson, and Geoffrey E Hinton. The toronto face dataset. Technical report, Technical Report UTML TR 2010-001, U. Toronto, 2010.

[11] Asja Fischer and Christian Igel. An introduction to restricted boltzmann machines. *CIARP*, 2012.

[12] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[13] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G.R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics Reports*, 810:1–124, 2019. A high-bias, low-variance introduction to Machine Learning for physicists.

[14] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4), Dec 2019.

[15] Y. Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Proceedings of the USSR Academy of Sciences*, 269:543–547, 1983.

[16] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[17] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019.

[19] Wei Zhang, Kazuyoshi Itoh, Jun Tanida, and Yoshiki Ichioka. Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Appl. Opt.*, 29(32):4790–4797, Nov 1990.

[20] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[21] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):100–108, 1979.

[22] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data mining and knowledge discovery*, 2(2):169–194, 1998.

[23] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, page 16–22, New York, NY, USA, 1999. Association for Computing Machinery.

[24] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[25] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

[26] Didier Fraix-Burnet, Marc Thuillard, and Asis K. Chattopadhyay. Multivariate approaches to classification in extragalactic astronomy. *Frontiers in Astronomy and Space Sciences*, 2:3, 2015.

[27] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[28] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 29–37, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[29] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[30] Adam Moss. Accelerated bayesian inference using deep learning. *Monthly Notices of the Royal Astronomical Society*, 496(1):328–338, May 2020.

[31] Akinori Tanaka and Akio Tomiya. Towards reduction of autocorrelation in hmc by machine learning, 2017.

[32] Daniel Levy, Matthew D. Hoffman, and Jascha Sohl-Dickstein. Generalizing hamiltonian monte carlo with neural networks, 2018.

[33] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks, 2017.

[34] Yuusuke Kataoka, Takashi Matsubara, and Kuniaki Uehara. Image generation using generative adversarial networks and attention mechanism. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–6, 2016.

[35] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. Accelerating science with generative adversarial networks: An application to 3d particle showers in multilayer calorimeters. *Physical Review Letters*, 120(4), Jan 2018.

[36] Aishik Ghosh. Deep generative models for fast shower simulation in ATLAS. *J. Phys. Conf. Ser.*, 1525(1):012077, 2020.

[37] Ulli Wolff. Critical slowing down. *Nuclear Physics B - Proceedings Supplements*, 17:93–102, 1990.

[38] N Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21(6):1087–1092, March 1953.

[39] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.

[40] Li Huang and Lei Wang. Accelerated monte carlo simulations with restricted boltzmann machines. *Physical Review B*, 95(3), Jan 2017.

[41] B. Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1 – 26, 1979.

[42] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[44] Ross Kindermann and Laurie Snell. *Markov random fields and their applications*, volume 1. American Mathematical Society, 1980.

[45] Scott E. Fahlman, Geoffrey E. Hinton, and Terrence J. Sejnowski. Massively parallel architectures for ai: NETL, Thistle, and Boltzmann Machines. *Proceedings of the Third AAAI Conference on Artificial Intelligence*, pages 109–113, 1983.

[46] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147 – 169, 1985.

[47] Geoffrey E. Hinton, Terrence J. Sejnowski, and David H. Ackley. *Boltzmann Machines: Constraint Satisfaction Networks That Learn*. Carnegie-Mellon University, Department of Computer Science, 1984.

[48] David E. Rumelhart and James L. McClelland. *Information Processing in Dynamical Systems: Foundations of Harmony Theory*, pages 194–281. MIT Press, 1987.

[49] Pankaj Mehta and David J. Schwab. An exact mapping between the variational renormalization group and deep learning. *CoRR*, abs/1410.3831, 2014.

[50] Giacomo Torlai and Roger G. Melko. Learning thermodynamics with boltzmann machines. *Phys. Rev. B*, 94:165134, Oct 2016.

[51] Alan Morningstar and Roger G. Melko. Deep learning the ising model near criticality. *Journal of Machine Learning Research*, 18:163:1–163:17, 2017.

[52] Satoshi Iso, Shotaro Shiba, and Sumito Yokoo. Scale-invariant feature extraction of neural network and renormalization group flow. *Phys. Rev. E*, 97:053304, May 2018.

[53] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G. R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab. A high-bias, low-variance introduction to machine learning for physicists. *CoRR*, abs/1803.08823, 2018.

[54] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, August 2002.

[55] Geoffrey Hinton. A practical guide to training restricted boltzmann machines, 2010.

[56] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002.

[57] Ruslan Salakhutdinov. Learning and evaluating boltzmann machines. *Technical Report UTML TR 2008 - 002*, 2008.

[58] Scott Kirkpatrick, C. D. Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220 4598:671–80, 1983.

[59] C. Jarzynski. Nonequilibrium equality for free energy differences. *Phys. Rev. Lett.*, 78:2690–2693, Apr 1997.

[60] Radford M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, April 2001.

[61] Magneto: 2D Ising model in C++. `https://github.com/s9w/magneto`. Accessed: Spring 2018.

[62] R. H. Swendsen and J. S. Wang. Nonuniversal critical dynamics in Monte Carlo simulations. *Phys. Rev. Lett.*, 58(2):86–88, January 1987.

[63] A. Sokal. *Monte Carlo Methods in Statistical Mechanics: Foundations and New Algorithms*, pages 131–192. Springer US, Boston, MA, 1997.

[64] U. Wolff. Critical slowing down. *Nucl. Phys. B (Proc. Suppl.)*, 18:93–102, 1990.

[65] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Phys. Lett. B*, 195:216–222, September 1987.

[66] A. D. Kennedy and B. J. Pendleton. Acceptances and autocorrelations in hybrid Monte Carlo. *Nucl. Phys. B*, 20:118–121, May 1991.

[67] M. Campostrini, P. Rossi, and E. Vicari. Monte Carlo simulation of $\mathbb{CP}^{N-1}$ models. *Phys. Rev. D*, 46(6):2647–2662, September 1992.

[68] L. Del Debbio, H. Panagopoulos, P. Rossi, and E. Vicari. Spectrum of confining strings in SU(N) gauge theories. *J. High Energy Phys.*, 2002(01), January 2002.

[69] L. Del Debbio, G. M. Manca, and E. Vicari. Critical slowing down of topological modes. *Phys. Lett. B*, 594:315–323, August 2004.

[70] J. Flynn, A. Jüttner, A. Lawson, and F. Sanfilippo. Precision study of critical slowing down in lattice simulations of the $\mathbb{CP}^{N-1}$ model, April 2015.

[71] C. Bonati and M. D'Elia. Topological critical slowing down: variations on a toy model. *Phys. Rev. E*, 98:013308, July 2018.

[72] B. Allés, G. Boyd, M. D'Elia, A. Di Giacomo, and E. Vicari. Hybrid Monte Carlo and topological modes of full QCD. *Phys. Lett. B*, 389:107–111, December 1996.

[73] S. Schaefer, R. Sommer, and F. Virotta. Critical slowing down and error analysis in lattice QCD simulations. *Nucl. Phys. B*, 845:93–119, April 2011.

[74] E. Vicari. Monte carlo simulation of lattice $\mathbb{CP}^{N-1}$ models at large N. *Phys. Lett. B*, 309:139–144, March 1993.

[75] U. Wolff. Collective Monte Carlo updating for spin systems. *Phys. Rev. Lett.*, 62(4):361–364, January 1989.

[76] D. Kusnezov and J. Sloan. Global demons in field theory. critical slowing down in the XY model. *Nucl. Phys. B*, 409:635–662, July 1993.

[77] H. G. Evertz, G. Lana, and M. Marcu. Cluster algorithm for vertex models. *Phys. Rev. Lett.*, 70:875–879, February 1993.

[78] N. Prokof'ev, I. S. Tupitsyn, and B. V. Svistunov. Exact, complete, and universal continuous-time worldline Monte Carlo approach to the statistics of discrete quantum systems. *J. Exp. Theor. Phys.*, 87(2), August 1998.

[79] J. Xu, Hui Li, and S. Zhou. An overview of deep generative models. *IETE Technical Review*, 32:131–139, December 2014.

[80] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models, March 2021.

[81] G. Torlai and R. G. Melko. Learning thermodynamics with Boltzmann machines. *Phys. Rev. B*, 94:165134, October 2016.

[82] L. Wang. Can boltzmann machines discover cluster updates?, February 2017.

[83] L. Huang and L. Wang. Accelerated Monte Carlo simulations with restricted Boltzmann machines. *Phys. Rev. B*, 95:035105, January 2017.

[84] A. Tanaka and A. Tomiya. Towards reduction of autocorrelation in HMC by machine learning, December 2017.

[85] Z. Liu, S. P. Rodrigues, and W. Cai. Simulating the Ising model with a deep convolutional generative adversarial network, October 2017.

[86] A. Morningstar and R. G. Melko. Deep learning the Ising model near criticality, August 2017.

[87] J. M. Urban and J. M. Pawlowski. Reducing autocorrelation times in lattice simulations with generative adversarial networks, November 2018.

[88] J. Singh, V. Arora, V. Gupta, and M. S. Scheurer. Generative models for sampling and phase transition indication in spin systems, June 2020.

[89] G. Kanwar, M. S. Albergo, D. Boyda, K. Cranmer, D. C. Hackett, S. Racanière, D. J. Rezende, and P. E. Shanahan. Equivariant flow-based sampling for lattice gauge theory, March 2020.

[90] K. A. Nicoli, C. J. Anders, L. Funcke, T. Hartung, K. Jansen, P. Kessel, S. Nakajima, and P. Stornati. On estimation of thermodynamic observables in lattice field theories with deep generative models, July 2020.

[91] D. Boyda, G. Kanwar, Racanière, D. J. Rezende, M. S. Albergo, K. Cranmer, D. C. Hackett, and P. E. Shanahan. Sampling using SU(N) gauge equivariant flows, August 2020.

[92] M. S. Albergo, D. Bodya, D. C. Hackett, G. Kanwar, K. Cranmer, S. Racanière, D. J. Rezende, and P. E. Shanahan. Introduction to normalizing flows for lattice field theory, January 2021.

[93] S. Lawrence and Y. Yamauchi. Normalizing flows and the real-time sign problem, January 2021.

[94] S. Foreman, X-Y. Jin, and J. C. Osborn. Deep learning Hamiltonian Monte Carlo, May 2021.

[95] D. Wu, R. Rossi, and G. Carleo. Unbiased Monte Carlo cluster updates with autoregressive neural networks, May 2021.

[96] E. G. Tabak and E. Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1), March 2010.

[97] E. G. Tabak and C. V. Turner. A family of nonparametric density estimation algorithms. *Commun. Pure Appl. Math.*, 66:145–164, November 2012.

[98] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows, May 2015.

[99] L. Dinh, D. Krueger, and Y. Bengio. NICE: Non-linear independent components estimation, October 2014.

[100] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP, May 2016.

[101] M. Lüscher. Trivializing maps, the Wilson flow and the HMC algorithm. *Commun Math Phys*, 293, November 2009.

[102] G. P. Engel and S. Schaefer. Testing trivializing maps in the Hybrid Monte Carlo algorithm. *Comput. Phys. Commun.*, 182:2107–2114, October 2011.

[103] K. L. Chung. *Markov Chains with Stationary Transition Probabilities*. Springer-Verlag, 2 edition, 1967.

[104] L. Tierney. Markov chains for exploring posterior distributions. *Ann. Statist.*, 22:1701–1762, 1994.

[105] S. Kullbach and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 1951.

[106] M. C. Gemici, D. J. Rezende, and S. Mohamed. Normalizing flows on Riemannian manifolds, November 2016.

[107] D. J. Rezende, G. Papamakarios, S. Racanière, M. S. Albergo, G. Kanwar, P. E. Shanahan, and K. Cranmer. Normalizing flows on tori and spheres, February 2020.

[108] P. Mehta and D. J. Schwab. An exact mapping between the variational renormalization group and deep learning, October 2014.

[109] M. Koch-Janusz and Z. Ringel. Mutual information, neural networks and the renormalization group. *Nature*, 14:578–582, March 2018.

[110] P. M. Lenggenhager, D. E. Gökmen, Z. Ringel, S. D. Huber, and M. Koch-Janusz. Optimal renormalization group transformation from information theory, September 2018.

[111] S. H. Li and L. Wang. Neural network renormalization group, February 2018.

[112] S. Efthymiou, M. J. S. Beach, and R. G. Melko. Super-resolving the Ising model with convolutional neural networks. *Phys. Rev. B*, 99:075113, February 2019.

[113] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, December 2014.

[114] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions, July 2018.

[115] G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation, May 2017.

[116] C. W. Huang, D. Krueger, A. Lacoste, and A. Courville. Neural autoregressive flows, April 2018.

[117] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models, October 2018.

[118] Thomas Müller, Brian Mcwilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling, August 2018.

[119] E. Hoogeboom, R. van den Berg, and M. Welling. Emerging convolutions for generative normalizing flows, January 2019.

[120] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. Cubic spline flows, June 2019.

[121] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. Neural spline flows, June 2019.

[122] C. Meng, Y. Song, J. Song, and S. Ermon. Gaussianization flows, March 2020.

[123] J. A. Gregory and R. Delbourgo. C2 rational quadratic spline interpolation to monotonic data. *IMA Journal of Numerical Analysis*, 3:141–152, April 1983.

[124] K. Hornik. Approximation capabilitities of multilayer feedforward networks. *Neural Networks*, 4:251–257, 1991.

[125] J. Köhler, L. Klein, and F. Noé. Equivariant flows: exact likelihood generative learning for symmetric densities, June 2020.

[126] J. Liu, Y. Qi, Z. Y. Meng, and L. Fu. Self-learning Monte Carlo method. *Phys. Rev. B*, 95:041101, January 2017.

[127] B. Efron. Bootstrap methods: another look at the jackknife. *Ann. Statist.*, 7(1):1–26, 1979.

[128] B Efron and R. Tibshirani. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science*, 1(1):54–77, 1986.

[129] S. Caracciolo and A. Pelissetto. Corrections to finite-size scaling in the lattice N-vector model for $N = \infty$. *Phys. Rev. D*, 58:105007, October 1998.

[130] I. Loshchilov and F. Hutter. Decoupled weight decay regularization, November 2017.

[131] I. Loshchilov and F. Hutter. SGDR: Stochastic gradient descent with warm restarts, August 2016.

[132] D. Masters and C. Luschi. Revisiting small batch training for deep networks, April 2018.

[133] R. Ge, F. Huang, C. Jin, and Y. Yuan. Escaping from saddle points - online stochastic gradient descent for tensor decomposition, March 2015.

[134] C. Zhang, Q. Liao, A. Rakhlin, B. Miranda, N. Golowich, and T. Poggio. Theory of deep learning III : Generalization properties of SGD. In *CBMM Memo No. 067*, July 2017.

[135] M. R. Wilson, J. Marsh Rossney, and L. Del Debbio. ANVIL (version 0.9), May 2021. `10.5281/zenodo.4792249`.

[136] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle,

A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[137] Z. Kassabov. Reportengine: A framework for declarative data analysis, February 2019. 10.5281/zenodo.2571601.

[138] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, October 1986.

[139] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[140] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle, 2015.

[141] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information, 2017.

[142] C-W. Huang, F. Ahman, K. Kumar, A. Lacoste, and A. Courville. Probability distillation: A caveat and alternatives, September 2018.

[143] R. K. Ellis, W. J. Stirling, and B. R. Webber. *QCD and Collider Physics*. Cambridge Monographs on Particle Physics, Nuclear Physics and Cosmology. Cambridge University Press, 1996.

[144] A. M. Stuart. Inverse problems: A bayesian perspective. *Acta Numerica*, 19:451–559, 2010.

[145] Madhu S. Advani, Andrew M. Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

[146] Rabah Abdul Khalek et al. Parton Distributions with Theory Uncertainties: General Formalism and First Phenomenological Studies. *Eur. Phys. J. C*, 79(11):931, 2019.

[147] Richard D. Ball, Emanuele R. Nocera, and Rosalyn L. Pearson. Nuclear Uncertainties in the Determination of Proton PDFs. *Eur. Phys. J. C*, 79(3):282, 2019.

[148] Richard D. Ball, Emanuele R. Nocera, and Rosalyn L. Pearson. Deuteron Uncertainties in the Determination of Proton PDFs. *Eur. Phys. J. C*, 81(1):37, 2021.

[149] J. Cruz-Martinez, S. Forte, and E.R. Nocera. Future tests of parton distributions. *Acta Physica Polonica B*, 52(3):243, 2021.

[150] Richard D. Ball, Valerio Bertone, Stefano Carrazza, Luigi Del Debbio, Stefano Forte, Patrick Groth-Merrild, Alberto Guffanti, Nathan P. Hartland, Zahari Kassabov, José I. Latorre, and et al. Parton distributions from

high-precision collider data. *The European Physical Journal C*, 77(10), Oct 2017.

[151] Stefano Carrazza and Juan Cruz-Martinez. Towards a new generation of parton densities with deep learning models. *The European Physical Journal C*, 79(8), Aug 2019.

[152] John C. Collins, Davison E. Soper, and George Sterman. *FACTORIZATION OF HARD PROCESSES IN QCD*, pages 1–91. WORLD SCIENTIFIC, 1989.

[153] G. Altarelli and G. Parisi. Asymptotic freedom in parton language. *Nuclear Physics B*, 126(2):298–318, 1977.

[154] Yuri L. Dokshitzer. Calculation of the Structure Functions for Deep Inelastic Scattering and e+ e- Annihilation by Perturbation Theory in Quantum Chromodynamics. *Sov. Phys. JETP*, 46:641–653, 1977.

[155] V N Gribov and L N Lipatov. Deep inelastic ep-scattering in a perturbation theory. *Yadern. Fiz. 15: No. 4, 781-807(Apr 1972).*, 1 1972.

[156] Richard D. Ball, Luigi Del Debbio, Stefano Forte, Alberto Guffanti, José I. Latorre, Juan Rojo, and Maria Ubiali. A first unbiased global nlo determination of parton distributions and their uncertainties. *Nuclear Physics B*, 838(1-2):136–206, Oct 2010.

[157] Valerio Bertone, Stefano Carrazza, and Nathan P. Hartland. Apfelgrid: A high performance tool for parton density determinations. *Computer Physics Communications*, 212:205–209, Mar 2017.

[158] Nathan Hartland. Proton structure at the lhc, 2014.

[159] Richard D. Ball, Stefano Carrazza, Luigi Del Debbio, Stefano Forte, Zahari Kassabov, Juan Rojo, Emma Slade, and Maria Ubiali. Precision determination of the strong coupling constant within a global pdf analysis. *The European Physical Journal C*, 78(5), May 2018.

[160] Ferran Faura, Shayan Iranipour, Emanuele R. Nocera, Juan Rojo, and Maria Ubiali. The strangest proton? *The European Physical Journal C*, 80(12), Dec 2020.

[161] Richard D. Ball et al. NNPDF4.0. *In preparation*, 2021.

[162] José Luis Bernal and John A. Peacock. Conservative cosmology: combining data with allowance for unknown systematics. *Journal of Cosmology and Astroparticle Physics*, 2018(07):002–002, Jul 2018.

[163] M. P. Hobson, S. L. Bridle, and O Lahav. Combining cosmological data sets: hyperparameters and bayesian evidence. *Monthly Notices of the Royal Astronomical Society*, 335(2):377–388, Sep 2002.

[164] Sjoerd Viktor Beentjes and Ava Khamseh. Higher-order interactions in statistical physics and machine learning: A model-independent solution to the inverse problem at equilibrium. *Physical Review E*, 102(5), Nov 2020.

[165] U. Wolff. Monte Carlo errors with less errors, June 2003.

[166] J. Dove, B. Kerns, R. E. McClellan, S. Miyasaka, D. H. Morton, K. Nagai, S. Prasad, F. Sanftl, M. B. C. Scott, A. S. Tadepalli, and et al. The asymmetry of antimatter in the proton. *Nature*, 590(7847):561–565, Feb 2021.

[167] Morad Aaboud et al. Precision measurement and interpretation of inclusive W^+ , W^- and Z/γ^* production cross sections with the ATLAS detector. *Eur. Phys. J.*, C77(6):367, 2017.

[168] M. Aaboud et al. Measurement of the Drell-Yan triple-differential cross section in pp collisions at $\sqrt{s} = 8$ TeV. *JHEP*, 12:059, 2017.

[169] Georges Aad et al. Measurement of the double-differential high-mass Drell-Yan cross section in pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector. *JHEP*, 08:009, 2016.

[170] Georges Aad et al. Measurement of $W^{\pm}$ and Z-boson production cross sections in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector. *Phys. Lett.*, B759:601–621, 2016.

[171] Morad Aaboud et al. Measurement of differential cross sections and W^+/W^- cross-section ratios for W boson production in association with jets at $\sqrt{s} = 8$ TeV with the ATLAS detector. *JHEP*, 05:077, 2018. [Erratum: JHEP 10, 048 (2020)].

[172] Georges Aad et al. Measurement of the $t\bar{t}$ production cross-section in the lepton+jets channel at $\sqrt{s} = 13$ TeV with the ATLAS experiment. *Phys. Lett. B*, 810:135797, 2020.

[173] Georges Aad et al. Measurements of top-quark pair differential cross-sections in the lepton+jets channel in pp collisions at $\sqrt{s} = 8$ TeV using the ATLAS detector. *Eur. Phys. J.*, C76(10):538, 2016.

[174] Morad Aaboud et al. Measurement of top quark pair differential cross-sections in the dilepton channel in pp collisions at $\sqrt{s} = 7$ and 8 TeV with ATLAS. *Phys. Rev. D*, 94(9):092003, 2016. [Addendum: Phys.Rev.D 101, 119901 (2020)].

[175] Morad Aaboud et al. Measurement of the inclusive jet cross-sections in proton-proton collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector. *JHEP*, 09:020, 2017.

[176] Georges Aad et al. Measurement of dijet cross sections in pp collisions at 7 TeV centre-of-mass energy using the ATLAS detector. *JHEP*, 1405:059, 2014.

[177] M. Aaboud, G. Aad, B. Abbott, J. Abdallah, O. Abdinov, B. Abeloos, S.H. Abidi, O.S. AbouZeid, N.L. Abraham, H. Abramowicz, and et al. Measurement of the cross section for inclusive isolated-photon production in pp collisions at s=13 tev using the atlas detector. *Physics Letters B*, 770:473–493, Jul 2017.

[178] Georges Aad et al. Comprehensive measurements of t-channel single top-quark production cross sections at $\sqrt{s} = 7$ TeV with the ATLAS detector. *Phys. Rev. D*, 90(11):112006, 2014.

[179] Morad Aaboud et al. Measurement of the inclusive cross-sections of single top-quark and top-antiquark t-channel production in pp collisions at $\sqrt{s} =$ 13 TeV with the ATLAS detector. *JHEP*, 04:086, 2017.

[180] Morad Aaboud et al. Fiducial, total and differential cross-section measurements of t-channel single top-quark production in pp collisions at 8 TeV using data collected by the ATLAS detector. *Eur. Phys. J. C*, 77(8):531, 2017.

[181] Serguei Chatrchyan et al. Measurements of differential jet cross sections in proton-proton collisions at $\sqrt{s} = 7$ TeV with the CMS detector. *Phys.Rev.*, D87:112002, 2013.

[182] Albert M Sirunyan et al. Measurement of the triple-differential dijet cross section in proton-proton collisions at $\sqrt{s} = 8\,\text{TeV}$ and constraints on parton distribution functions. *Eur. Phys. J. C*, 77(11):746, 2017.

[183] A. M. Sirunyan et al. Measurement of the inclusive $\mathrm{t\bar{t}}$ cross section in pp collisions at $\sqrt{s} = 5.02$ TeV using final states with at least one charged lepton. *JHEP*, 03:115, 2018.

[184] Albert M Sirunyan et al. Measurement of double-differential cross sections for top quark pair production in pp collisions at $\sqrt{s} = 8$ TeV and impact on parton distribution functions. *Eur. Phys. J.*, C77(7):459, 2017.

[185] Albert M Sirunyan et al. Measurement of differential cross sections for the production of top quark pairs and of additional jets in lepton+jets events from pp collisions at $\sqrt{s} = 13$ TeV. *Phys. Rev.*, D97(11):112003, 2018.

[186] Albert M Sirunyan et al. Measurements of $\mathrm{t\bar{t}}$ differential cross sections in proton-proton collisions at $\sqrt{s} = 13$ TeV using events containing two leptons. *JHEP*, 02:149, 2019.

[187] Serguei Chatrchyan et al. Measurement of the Single-Top-Quark t-Channel Cross Section in pp Collisions at $\sqrt{s} = 7$ TeV. *JHEP*, 12:035, 2012.

[188] Vardan Khachatryan et al. Measurement of the t-channel single-top-quark production cross section and of the $\mid V_{tb} \mid$ CKM matrix element in pp collisions at $\sqrt{s}$= 8 TeV. *JHEP*, 06:090, 2014.

[189] Albert M Sirunyan et al. Cross section measurement of t-channel single top quark production in pp collisions at $\sqrt{s} = 13$ TeV. *Phys. Lett. B*, 772:752–776, 2017.

[190] Roel Aaij et al. Measurement of the forward Z boson production cross-section in pp collisions at $\sqrt{s} = 13$ TeV. *JHEP*, 09:136, 2016.