Machine Learning Methods for the Simulation of Lattice Field Theories

A thesis submitted in fulfilment of the requirements

for the degree of Doctor of Philosophy

by

Ankur Singha

18109261

Department of Physics

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

June 2023

Certificate

It is certified that the work contained in this thesis entitled **" Machine Learning Methods for the Simulation of Lattice Field Theories"** by **Ankur Singha 18109261** has been carried out under my supervision and that it has not been submitted elsewhere for a degree.

Prof. Dipankar Chakrabarti
Professor
Department of Physics
Indian Institute of Technology Kanpur

Dr. Vipul Arora
Associate Professor
Department of Electrical Engineering
Indian Institute of Technology Kanpur

Declaration

This is to certify that the thesis titled **" Machine Learning Methods for the Simulation of Lattice Field Theories"** has been authored by me. It presents the research conducted by me under the supervision of **Prof. Dipankar Chakrabarti** and **Dr. Vipul Arora**.

To the best of my knowledge, it is an original work, both in terms of research content and narrative, and has not been submitted elsewhere, in part or in full, for a degree. Further, due credit has been attributed to the relevant state-of-the-art and collaborations with appropriate citations and acknowledgments, in line with established norms and practices.

Ankur Singha
18109261
Roll No. 18109261
Physics Department
Indian Institute of Technology Kanpur

Synopsis

Name of the student: **Ankur Singha**

18109261 Roll No: **18109261**

Degree for which submitted: **PhD** Department: **Physics Department**

Thesis title: **Machine Learning Methods for the Simulation of Lattice Field Theories**

Thesis supervisors: **Prof. Dipankar Chakrabarti** and **Dr. Vipul Arora**

Month and year of thesis submission: **June 2023**

The Lattice Quantum Chromodynamcs (LQCD) is a non-perturbative method, proposed by K.G. Wilson for solving QCD numerically. Lattice QCD is based on Feynman Path Integral Formalism in Euclidean space, where the spacetime is discretized into a grid of hypercube. The path integral is numerically estimated using Monte Carlo methods on a Computer, where physical observables are estimated as ensemble average over a large number of simulated lattice configurations. Lattice QCD has successfully calculated various properties of hadron, their mass, Parton distribution functions, the behaviour of quarks and gluon under extreme conditions such as high temperature and density, physics of light nuclei, the muon magnetic dipole moment etc. Lattice QCD is currently playing an important role in determining anomalies in B physics that emerges from the LHC, exploring the rare Kaon decay, and hadronic contribution to muon magnetic moment etc.

Although Lattice QCD has made good progress, it requires huge computational resources. In Lattice QCD, Monte Carlo simulations involve generating many lattice field configurations and averaging them to obtain physical quantities. These simulations are computationally intensive, and the statistical errors associated with them can be significant,

requiring a large number of samples to reduce statistical uncertainties. To obtain continuum results, it is necessary to take the limit of vanishing lattice spacing (the continuum limit). However, this requires performing calculations at multiple lattice spacing and extrapolating the results into continuum limits, increasing the computational effort. The Monte Carlo simulation cost increases rapidly as we move towards the critical region of the lattice theory, i.e. simulation cost depends on the parameter value of the lattice theory at which the lattice samples are generated. This is due to the increase in correlation amongst the samples in the Markov Chain as we move towards the critical point. The correlation measure is given by integrated autocorrelation time, which diverges at the critical point. This problem is known as the Critical Slowing Down (CSD) in Monte Carlo simulations. As the autocorrelation time becomes longer, the statistical uncertainties of observables increase, i.e. it leads to a biased estimation of observables.

In ML literature, transfer learning techniques have been developed to transfer knowledge across domains efficiently. This transfer enables initializing the model efficiently for the new domain. Likewise, conditional generative models can regress across continuous parameters, thereby transferring knowledge across parameter values. In this thesis, we propose Machine learning (ML) based Generative models for efficient simulation of Lattice Field Theory. Generative models work by learning the underlying patterns and features of the training data and then using this knowledge to generate new data. It learns the underlying probability distribution of a dataset to generate new, similar samples.

We propose a Generative Adversarial Network (GAN), an implicit generative model for sampling lattice fields near the critical region. Our approach is based on the fact that the non-critical samples in Lattice Field theory can be generated at a low cost and used to train a conditional GAN (C-GAN) to learn a conditional distribution across the action parameter of the lattice theory. The trained C-GAN model can be interpolated/extrapolated near the critical region to generate new samples. We test our approach on the lattice Gross-Neveu model in 2 dimensions. We observe no correlation amongst the samples from the C-GAN as GAN inherently generates uncorrelated samples, thus reducing the critical slowing down problem. Additionally, we conduct an empirical investigation

on various observables. GAN can't provide density for the generated samples; therefore, we can not ensure the exactness of sampling from the target distribution. Hence, we propose an MCMC-based sampler using a conditional Normalizing Flow model (C-NF). The C-NF model is trained on the non-critical region; however, in this case, the model is not directly used for sampling. After interpolation, use the model as a proposal sampler for constructing a Markov Chain. We use an independent Metropolis Hasting (MH) algorithm to make an exact sampler. Once training is over, we can generate a Markov chain for any parameter in the non-training regions. We test this proposed method on a simpler problem such as ϕ^4 theory in 2 dimensions. We also present an extension of the above ideas for sampling Lattice Gauge Theories. We propose an equivariant conditional flow-based model for sampling a $U(1)$ gauge theory in 2 dimensions.

Chapter 2 briefly overviews the Lattice Field Theory and the basics of Monte Carlo simulation methods. We present the discretization of a generic Quantum Field Theory (QFT) on Euclidean spacetime. The Hybrid Monte Carlo method has been thoroughly discussed.

Chapter 3 begins with an introduction to Machine Learning and its various classifications. Then, we define generative models and discuss generative models, such as Generative Adversarial Networks (GAN) and Normalizing Flow (NF).

In Chapter 4, we discuss the application of GAN for the simulation of the lattice Gross-Neveu model in 2 dimensions. GAN has never been used for lattice fermions simulations. We explain how to build a Conditional GAN (C-GAN), which can learn a conditional distribution over the action parameter of the Gross-Neveu model. The trained model, which is a conditional model, can be interpolated near the critical point for generating lattice samples. We present the different results, including observable like - mean absolute magnetization $\langle|\bar{\sigma}|\rangle$, susceptibility χ etc., which are calculated on the critical regions from the conditional GAN model (C-GAN) and HMC simulations.

In Chapter 5, we build a flow-based model for exact sampling for ϕ^4 theory in 2 dimensions. The GAN has one limitation: it can not provide the density of the generated sample. Therefore, we have shifted to a flow-based model which can sample as well as estimate the

density of generated samples. We discuss a conditional flow model (C-NF) which learns a conditional distribution and constructs a Markov Chain using Metropolis Hasting (MH) algorithm, where proposals are generated from the C-NF model.

In Chapter 6, we extend the application of C-NF to $U(1)$ gauge theory in two dimensions without fermions. We train an Equivariant conditional flow model to learn a conditional distribution over the $U(1)$ gauge parameter β for small values. Then we extrapolate the model for larger β values and calculate topological quantities. We provide a re-trainable method where the model utilizes its own samples to improve performances far from the training region.

In Chapter 7, we provide a brief summary of the thesis and conclude with some prospects for the future.

Acknowledgements

Standing at the culmination of this arduous journey, I am humbled and filled with gratitude for many individuals and entities. Without their support, encouragement, cooperation and guidance this would not have been possible.

First and foremost, I would like to thank my supervisor, Prof. Dipankar Chakrabarti and co-supervisor Dr Vipul Arora for encouraging me to try to be an independent researcher. It has been a valuable experience for me to work under their guidance for the last few years. Their invaluable guidance, discussions, and patient support have influenced not only the development of this thesis but also my academic and personal growth.

I am grateful to my doctoral committee members Prof. Joydeep Chakrabortty, Dr Narayan Rana and Prof. Kaushik Bhattacharya for their insightful suggestions, for monitoring the progress of my research and for providing me with constructive input every semester. The interaction and academic activities within the HEP group of IIT Kanpur are also worth mentioning. The list of acknowledgements further extends to the technical section and all the office staff of the physics department for handling the technical as well as the official processes.

I hold a deep admiration for Alolika, Ansuma Narzari, Shreya Madam and Jutika Boro for their exceptional guidance and support during my PhD journey. Their invaluable insights, constructive suggestions, and mentorship were instrumental in shaping my research and personal growth. Additionally, they played a crucial role in helping me maintain a balanced and healthy lifestyle.

I extend my gratitude to my senior Dr Raj Kishore and Labmates Poonam, Abhijeet, Bhimsen, and Sumit, with whom I had many fascinating discussions.

My friends both within and beyond IITK have played a crucial role in my life over the past five years, and words cannot adequately express my gratitude towards them. I consider myself fortunate to have friends like Bhaskarya, Ankur Sarma, Sharat, Tapan, Hemanta, Kumar, Kaushik, Jyotismrita, Mukesh, Prodyut, Gautam singha, Suman Karn, Suman Chakrabarti, Soumydeep, Santu, suprodip, Rinku, Himadri, Swaswat, Kishore, Ashis, Nikhil, Kirtiman, Swati, Ayesha, Kalyani, Prateesh, Rohan, vinayank,Harishyam, Sushanta, Luv Kumar, Mir, Bidyut and many others.

I am indebted to my teachers for imparting their enriching knowledge during my schooling, B.Sc and M.Sc tenure.

I would like to express heartfelt gratitude to my parents, for their unconditional love, unwavering support and sacrifices without which the journey of my higher education would not have even started. Also, I convey special thanks to my siblings Narayan, Apsara, Nandalal, and Amori for their understanding and encouragement. I also thank Nripen, Gupta, Gita, Runu, Ankita, Richel, Nono, Jinia, Rishik, Ananya, Pranjit and other family members. They have been a constant source of inspiration.

I am immensely grateful to our institute IIT Kanpur for providing me with a nurturing academic environment and resources that have been crucial in conducting my research.

Last but not least, I extend my deepest appreciation to all the anonymous individuals who generously volunteered their time and shared their knowledge to contribute to my research.

Contents

List of Figures

List of Tables

List of Publications

Publications from Thesis

1. Generative learning for the problem of critical slowing down in lattice Gross-Neveu model - Ankur Singha, Dipankar Chakrabarti, Vipul Arora.
   SciPost PhysCore5, 052, (2022).

2. Conditional normalizing flow for Markov chain Monte Carlo sampling in the critical region of lattice field theory - Ankur Singha, Dipankar Chakrabarti, Vipul Arora.
   Phys. Rev. D107, 014512, (2023).

3. Sampling U(1) gauge theory using a re-trainable conditional flow-based model - Ankur Singha, Dipankar Chakrabarti, Vipul Arora.
   arXiv: 2306.00581.

Dedicated to my parents.

Nirmal Singha & Pingala Singha

Chapter 1

Introduction

It took physicists a long journey via experimentation and theory to uncover the fundamental building block of the universe, which led to the Standard Model (SM) of Particle Physics. The Standard Model is a relativistic Quantum Field Theory (QFT) which describes fundamental particles and their interactions. It is the most widely accepted model that explains fundamental forces such as Electromagnetic, Weak, and Strong, except Gravity. It is a highly successful theory that has been extensively tested through experiments and precisely predicted various phenomena. The last and the most elusive particle predicted by the SM was discovered at Large Hadron Collider (LHC) in 2012.

The Standard Model of particle physics is a Gauge Theory with symmetry group $SU(3)_C \times SU(2)_L \times U(1)_Y$. At high energies, the weak and electromagnetic forces are unified and represented by the symmetry $SU(2)_L \times U(1)_Y$, which is known as Electroweak theory, and the $SU(3)$ symmetry group corresponds to the Quantum Chromodynamics (QCD) which describes the dynamics of quarks and gluons. In QCD, quarks are in the fundamental representation of $SU(3)$, and the eight vector bosons (gluon) are in the adjoint representation. For the first time, the quark was introduced in 1964 in the quark model [1] by Gellmann and George Zweig. It was developed as a classification scheme for hadrons using the representations of the $SU(3)$ colour group. They proposed that hadrons are made up of quarks which come in six flavours - up, down, strange, charm, bottom and top. Hadrons include both baryons, consisting of three quarks and mesons, consisting of a quark and an antiquark. One of the great successes of the quark model is the prediction of Ω^- baryon [2]. However, a free quark had never been observed in any experiment. The quark model could not explain the confinement of quarks in the hadrons. Yang-Mills (YM) theory and generalization of QED with YM theory gave birth to QCD, a non-abelian gauge theory.

Asymptotic freedom is a fundamental property of QCD [3] [4]. The QCD coupling parameter (α), which is large at low energy, decreases with the energy and becomes weak at very high energy. This explains the confined nature of quarks at the low energy scale. In the high energy scale where the QCD coupling constant has small values, perturbative methods successfully make predictions. However, at low energy, the perturbative method fails where most of the hadron physics lies. So, we need non-perturbative methods in quantum field theory to explore the low-energy region of QCD. Lattice QCD is the non-perturbative method first proposed by K.G. Wilson in 1974 [5] and explained the confinement nature. Lattice QCD is rooted in the Feynman Path Integral formalism within Euclidean space, where the spacetime is discretized into a grid of hypercube. The path integral is numerically estimated using Monte Carlo methods on a Computer, where physical observables are estimated as ensemble average over a large number of simulated lattice configurations.

Lattice QCD has been successful in calculating various properties of hadron, their mass, Parton distribution functions, the behaviour of quarks and gluon under extreme conditions such as high temperature and density [6, 7], physics of light nuclei [8, 9], $g-2$, the Muon magnetic dipole moment [10, 11]. Advancements in computing power and available computational resources have greatly facilitated lattice QCD progress. The availability of powerful supercomputers and high-performance computing (HPC) facilities has been instrumental in advancing lattice QCD. These facilities provide researchers with substantial computational resources, including large memory capacities, high-speed interconnections, and parallel processing capabilities. GPUs also offer high parallel computing power, enabling researchers to accelerate lattice QCD simulations. Furthermore, different lattice actions have been developed to reduce lattice artefacts and improve the agreement with the continuum theory. Symanzik improvement is a systematic approach to removing discretization errors by adding correction terms to the lattice action. Multigrid techniques have been applied to lattice QCD simulations to accelerate the convergence of iterative solvers in solving the lattice equations. These advancements allow researchers to perform calculations with larger lattice volumes, finer lattice spacing, and more sophisticated algorithms, thereby reducing discretization errors and enhancing the accuracy of results.

Despite the significant progress made in Lattice QCD, it continues to demand substantial computational resources. Moreover, there is a pressing need to increase the statistical precision by several orders of magnitude from the current state. Then only Lattice QCD can precisely determine the phenomenologically demanding quantities like the hadronic contribution to the anomalous magnetic moment of the muon [11, 12], understanding of anomalies in B physics [13] which emerges from the LHC and also exploring the rare Kaon

decay [14] apart from understanding the complicated internal structure of the hadronic bound states. These calculations will be critical for interpreting results from the Fermilab Muon $g-2$ experiments [10, 11].

Lattice methods have applications in various areas, such as statistical systems, condensed matter systems, and other Quantum Field Theories (QFT). In Lattice Field Theory (LFT), Markov Chain Monte Carlo (MCMC) methods are commonly employed to sample the probability distribution defined by the lattice system's action or Hamiltonian. However, in all cases, the computational cost of Monte Carlo simulations increases significantly as we approach the critical region of the lattice system. In other words, the simulation cost depends on the parameter value of the theory at which the lattice samples are generated. Most MCMC algorithms perform well in the non-critical region of the lattice theory since there is typically a low correlation between the generated samples. However, simulations near the critical region require larger lattice volumes and finer lattice sizes, which, in turn, increase the correlation among the samples in the Markov Chain. The level of correlation is quantified by the integrated autocorrelation time, which diverges precisely at the critical point. This well-known issue is referred to as "Critical Slowing Down"[15, 16]. This creates problems in simulation and observable estimation, which forces us to increase statistics or ensemble size; otherwise, it may lead to a biased estimation from the simulation. The Critical Slowing Down depends on both the Monte Carlo algorithm as well the observable we try to estimate. There has been some effort to reduce critical slowing down by tuning existing MCMC algorithm and some practical implementation has been proposed[17–27]. Several MCMC algorithms have been developed for specific theories [28–32, 32–38] to reduce the effect of critical slowing down. They reduce the critical slowing down problem for many statistical problems but are not directly applicable to LFT like lattice QCD. Consequently, critical slowing down remains a major obstacle in accurately estimating physical observables within numerous lattice field theories[39–41].

Machine Learning (ML) has gained significant popularity and is being extensively applied across various fields of physics to tackle complex theoretical problems, analyze experimental data, and make predictions. An illustrative example is its utilization in particle physics, where ML techniques are employed to analyze large volumes of data produced by high-energy experiments [42–45]. ML algorithms aid in particle identification and classification, anomaly detection[46], and predictive modelling. In the domain of LFT, ML-based methods are utilized for sampling lattice distributions. Furthermore, ML-based Generative models have demonstrated immense potential in efficiently addressing the challenge of critical slowing down. Considerable efforts have been dedicated to developing ML-based

generative models [47–59] to reduce critical slowing down in statistical and LFT. Flow-based generative models have been found to be effective in avoiding the problem of critical slowing down for specific theories [60–66]. Flow-based models serve as generative models utilized to sample a probability distribution that is defined by a lattice action. These models are constructed and trained to acquire knowledge about the probability distribution and subsequently generate lattice samples. In LFT, the current flow-based methods[60–66], a Markov chain utilizes a flow model that is trained for a specific action parameter value. Similarly, traditional MCMC methods commence the Markov chain from a random state for each action parameter value. This approach proves inefficient when estimates are required for numerous action parameter values. Simulations at multiple parameter values are crucial for studying the critical properties of a statistical lattice system and achieving the continuum limit of a LFT. For example, estimating the free energy necessitates lattice ensembles at discrete steps in the parameter space of the theory[67]. Consequently, training distinct flow models for each parameter value escalates the simulation cost. Moreover, flow-based methods that employ online training[61, 62], i.e., learning the flow models while generating the samples, instead of pre-training with samples from the training data, can be prone to instability and mode collapse.

The primary focus of this thesis approached from a machine learning (ML) perspective, is the question of whether the knowledge obtained from the non-critical region of a lattice system (characterized by low autocorrelation) can be leveraged to generate lattice samples in the critical region of the lattice theory. In the field of ML, transfer learning techniques have been developed to facilitate efficient knowledge transfer across different domains. This transfer enables the initialization of models in a more effective manner for new domains. To address this task, we propose the utilization of machine learning-based generative models.

Generative models operate by learning the underlying patterns and features present in the training data and subsequently employing this acquired knowledge to generate new data. These models have been successfully employed in generating realistic images, videos, and audio that are virtually indistinguishable from real-world data. Generative models can learn the data distribution either explicitly or implicitly. Explicit generative models such as Normalizing flow, also known as density models, explicitly learn the probability distribution of the training data. On the other hand, implicit generative models such as Generative Adversarial Networks (GAN) do not explicitly learn the probability distribution of the data but still possess the ability to generate samples that resemble the training data. We have utilised both kinds of generative models for the sampling task in LFT.

We have introduced a GAN as an implicit generative model designed to sample lattice fields in the vicinity of the critical region of the lattice Gross-Neveu model in 1+1 dimensions. Our approach is based on the fact that in the non-critical region of the LFT, it is feasible to generate lattice samples at a low computational cost using simulation algorithms such as Hybrid Monte Carlo (HMC). These generated samples are then utilized for training a GAN that is conditioned on the action parameter, enabling the learning of a conditional distribution across the action parameter space of the lattice theory. Subsequently, we employ interpolation techniques to extend the trained conditional GAN (C-GAN) to generate new samples near the critical region. By leveraging the GAN framework, each sample is generated starting from a random lattice variable, ensuring that there is no correlation among the samples produced by the C-GAN.

A limitation of the Generative Adversarial Network (GAN) model is that it does not provide a probability density estimation for the generated samples. Hence, we propose an MCMC-based sampler that utilizes a Normalizing Flow to construct a Markov Chain for ϕ^4 theory in 1+1 dimensions. Building upon our previous work, we employ the C-NF model to interpolate in the critical region and utilize our model as a proposal sampler for constructing a Markov Chain. To ensure exact sampling, we employ the independent Metropolis-Hastings (MH) algorithm. Once the training is completed, we can generate a Markov chain for any parameter in the non-training regions, enabling lattice generation at multiple parameter values.

Furthermore, we extend the concept of conditional flow models to simulate U(1) gauge theory (1+1) dimensions. Our objective is to generate lattice samples in scenarios where conventional MCMC algorithms fail due to topological freezing. To achieve this, we generate training samples from a parameter region characterized by low autocorrelation in the topological charge. Subsequently, we train an equivariant flow model and generate samples for different action parameter values to reduce autocorrelation in topological quantities.

The following organizational structure is employed in the thesis-

In Chapter 2 a concise introduction to LFT and the fundamental principles of Monte Carlo simulation techniques will be provided.will give a brief introduction to LFT and the fundamental principles of Monte Carlo simulation techniques. We will present the process of discretizing a generic Quantum Field Theory (QFT) on Euclidean spacetime. Additionally, we will provide a concise overview of the Lattice Gauge Theory, focusing on the development of the SU(N) gauge symmetric action. Subsequently, we will examine

different MCMC algorithms commonly employed in LFT simulations, with special emphasis on the comprehensive discussion of the Hybrid Monte Carlo method. Lastly, we will explore the issue of critical slowing in Monte Carlo simulations and its implications for lattice simulations.

Chapter 3 commences by providing an overview of Machine Learning and its diverse classifications. Subsequently, we present the concept of generative models and explore different instances, including Generative Adversarial Networks (GAN), Variational-Auto-encoder (VAE), Boltzmann Machine, and Normalizing Flow (NF). Detailed attention is devoted to GAN and NF, as these models are employed extensively in our research endeavours for this thesis.

In Chapter 4, we will present our generative model-based ML approach for sampling LFT. We discuss the application of GAN for the simulation of the lattice Gross-Neveu model in 2 dimensions. We will go over how to build a Conditional GAN for efficiently sampling a distribution specified by the Gross-Neveu action, particularly in the critical region of the theory. We will present the results we obtain for critical regions from a conditional GAN model (C-GAN).

In Chapter 5, we will build a flow-based model for exact sampling for ϕ^4 theory in 2 dimensions. We will discuss a conditional flow model (C-NF) which learns a conditional distribution across the parameter value of the lattice theory. Then, we give the training and sampling procedure for the C-NF model and present the result in the critical region of the theory.

In Chapter 6, we extend the application of C-NF from scalar theory to gauge Theory. We work with pure $U(1)$ gauge theory in two dimensions without fermions. For the gauge field sampler, the C-NF architecture must be built such that it generates samples from a gauge symmetric distribution, otherwise, it leads to inefficient generation and a low acceptance rate in the MH step. So, we use an architecture where the actual flow transformation occurs in the gauge symmetric quantity, which has a one-to-one correspondence to the lattice link variable. This kind of flow is also known as equivariant flow. As in the previous case, we train a conditional model to learn a conditional distribution over the $U(1)$ gauge parameter β for small values. Then we extrapolate the model for larger β values. For β values far from the training region the acceptance rate in the MH step may decrease. To overcome this, we provide a re-trainable method where the model utilizes its own samples to improve performances far from the training region. We discuss the training, and sampling procedure and compare the observables like the Wilson loop, Topological

charge etc. against HMC simulations. In Chapter 7, we provide a brief summary of the thesis and conclude with some prospects for the future.

Chapter 2

Lattice Field Theory and Monte Carlo Simulations

Lattice field theory is a computational technique used in particle physics and other areas to study the behaviour of quantum field theories, especially in a nonperturbative regime where the coupling parameter becomes strong. Until 1974, all predictions of QFT like QCD were calculated using perturbative methods where the coupling parameter is weak and hence predictions were restricted to weak coupling regimes only. In 1974 Kenneth Wilson [5] opened a new way to the study of non-perturbative phenomena using numerical methods. In this approach, a continuous space-time is discretized into a lattice, allowing calculations to be performed using powerful numerical techniques. It is based on the idea that physical observables can be calculated as averages over a large number of lattice configurations. These configurations are generated using a computer algorithm such as the MCMC that samples from the space of all possible field configurations. For detailed discussions on LFT, please see [68–71]. A few basic terms of LFT are also discussed in Appendix B for quick reference.

In a quantum field theory, the Green functions are the objects which contain most of the information necessary for making any prediction from the theory. Feynman Path Integral formalism [72, 73] estimates n-point Green functions in QFT as integrals over classical field configurations. The N-point correlation function in QFT can be written as

$$\left\langle 0|T[\hat{O}_1...\hat{O}_N]|0\right\rangle = \frac{\int \mathcal{D}\phi O_1[\phi],...,O_N[\phi]e^{iS(\phi)}}{Z}, \tag{2.1}$$

where,

$$Z = \int \mathcal{D}\phi e^{iS(\phi)}. \tag{2.2}$$

First of all, there is no guarantee of the integral's convergence because it has oscillating integrands due to the imaginary term in the exponential. Secondly, since the theory has infinite degrees of freedom, the integral measure $\mathcal{D}\phi$ is difficult to define. The first problem is readily handled if we make a Wick rotation and move to the imaginary time.

$$\Big\langle O_1 ... O_N \Big\rangle = \frac{1}{Z} \int \mathcal{D}\phi O_1[\phi], ..., O_N[\phi] e^{-S_E(\phi)}. \tag{2.3}$$

In the imaginary time formulation, $t = i\tau$ is used, where the distinction between the time and space components disappears and the metric becomes Euclidean. The Euclidean functional integral allows us to perform importance sampling in the Monte Carlo simulations.

2.1 Discretiztion of Euclidean Space:

We discretize the d dimensional Euclidean space-time into a d dimensional hyper-cubic grid with lattice spacing a [5]. We use $\hat{\mu}$ to denote different unit vectors in d different directions. A lattice point in the hypercubic lattice can be defined as

$$\Gamma = \Big\{ x : x = \sum_{\mu=1}^{d} a n_\mu \hat{\mu}; \quad n_\mu \in \mathcal{Z}. \tag{2.4}$$

For example, a scalar field on the lattice can be written as $\phi(x); \quad x \in$ hypercube.

Upon discretization the spacetime integral becomes

$$\int d^d x \longrightarrow \sum_x a^d$$

And the integral measure becomes well-defined in a finite lattice volume

$$\int \mathcal{D}\phi(x) = \Pi_x d\phi(x).$$

The functional integral on the lattice is as follows

$$Z = \int \mathcal{D}\phi(x) \exp(-S_E(\phi(x))). \tag{2.5}$$

where, $x \in$ lattice hypercube. An observable on the lattice can be calculated as

$$\left\langle O \right\rangle = \frac{1}{Z} \int \mathcal{D}\phi \; O[\phi] e^{-S_E(\phi)}. \tag{2.6}$$

Evaluating the Equation (2.6) is equivalent to sampling lattice field configurations ϕ_i according the distribution

$$P(\phi_i) = \frac{e^{-S(\phi_i)}}{Z}. \tag{2.7}$$

In Monte Carlo simulation we generate samples according to the distribution defined in Equation (2.7) and estimate observables as

$$\left\langle O \right\rangle \approx \frac{1}{N} \sum_{i=1}^{N} O[\phi_i]. \tag{2.8}$$

However, upon discretization, the lattice spacing sets a momentum cutoff. Therefore direct estimation of observables will not match with predictions of continuum theory. This fact becomes evident upon transforming the field $\phi(x)$ into Fourier space,

$$\tilde{\phi}(p) = a^d \sum_x e^{-p.x} \phi(x)$$

In momentum space, the fields exhibit periodicity

$$p_\mu \longrightarrow p_\mu + \frac{2\pi}{a}.$$

And

$$-\frac{\pi}{a} < p_\mu \leq \frac{\pi}{a}.$$

So, LFT provides momentum regularisation naturally. Therefore, one should ideally estimate lattice results near the continuum limit of the lattice theory. To achieve this, LFT simulations are performed at different values of the lattice spacing, typically using a range of lattice spacings. By studying the behaviour of observables as the lattice spacing decreases, it is possible to extrapolate the results to the continuum limit. We have to work not only in finite lattice spacing but also need to work in finite lattice volume. Let us consider a lattice hypercube with lengths

$$L_1 = L_2 = ... = L_{d-1} = L$$

in the space direction and $L_d = T$ in the time direction. Therefore, we have to work in

a finite lattice of volume $V = L^{d-1}T$. The thermodynamic limit in LFT refers to the procedure of taking the volume of the lattice to infinity while keeping the lattice spacing fixed. Taking the thermodynamic limit is essential because it allows for the removal of finite volume effects. We apply boundary conditions on the finite lattice such as periodic boundary conditions, anti-periodic boundary conditions or open boundary conditions etc and the behaviour of the system can be influenced by such boundary conditions.

2.1.1 Analogy to a Statistical system:

Let us consider a statistical mechanical spin system with Hamiltonian $H(\mathbf{s})$ where $\mathbf{s}$ is a spin variable on some lattice site. In the canonical ensemble, the expectation of the observable can be written as

$$< O >= \frac{1}{Z} \sum_{\{\mathbf{s}\}} e^{\beta H[\mathbf{s}]} O(\mathbf{s}), \tag{2.9}$$

Where, the partition function is

$$Z = \sum_{\{\mathbf{s}\}} e^{\beta H[\mathbf{s}]}, \tag{2.10}$$

$\beta = \frac{1}{k_B T}$, T is the heat bath temperature and the sum $\{\mathbf{s}\}$ is over all possible spin configuration.

If we compare Equation (2.5) with Equation (2.9) we find the weighted action $exp(-s_E)$ represent the $exp(-\beta H)$ in the statistical system. The sum over field configuration is equivalent to the sum over lattice field configurations. The importance of connection is necessary as it allows us to use already existing numerical methods in statistical mechanics for the simulation of the lattice field.

2.1.2 Continuum Limit:

In lattice theory we have to remove the ultra-violet cutoff, $\frac{1}{a}$ which is equivalent to taking the continuum limit $a \to 0$. If we want to estimate the mass spectrum of the continuum field theory by studying an LFT then we can compute an appropriate correlator function in large Euclidean time. The lowest mass m corresponds to the largest correlation length. If $\hat{m}$ is the corresponding mass in the lattice unit then in the continuum limit it must

vanish i.e. the corresponding correlation length in lattice unit ξ must diverge [74]. So,

$$a \longrightarrow 0 \implies \hat{\xi} \longrightarrow \infty$$

or

$$m << \frac{1}{a}.$$

But this corresponds to a critical point in the lattice theory and a second-order phase transition in a statistical system. Therefore, the lattice theory must possess a critical point which gives physical observable of the corresponding theory.

Now, studying the lattice system near the critical region corresponds to tuning the action parameter of the lattice theory. For some particular value of the action parameter, we will reach the critical region. In a statistical system, we know that the critical point depends on the temperature T. The correlation length in the lattice unit depends on the parameter or coupling (g) of the action i.e. $\hat{\xi} = \hat{\xi}(g)$. A particular value of $g = g^*$ will corresponds to a critical point where $\hat{\xi}(g^*) \longrightarrow \infty$. let us consider a simple case where g is dimensionless. If Θ_{phy} is the physical observable of mass dimension d and $\hat{\Theta}$ is the corresponding lattice quantity then the existence of a critical point implies,

$$\Theta(g, a) = \left(\frac{1}{a}\right)^d \hat{\Theta}(g). \tag{2.11}$$

to reach a finite quantity for $a \to 0$. The action parameter must be a function of the action parameter $g(a)$. We can tune it so as to reach the critical point ($g(a) \to g^*$) and get a finite physical estimate-

$$\theta(g(a), a) \longrightarrow \theta_{phy}. \tag{2.12}$$

Ideally one should take the infinite volume limit or thermodynamic limit, $L \to \infty, T \to \infty$ and then take the limit $a \to 0$. Taking the continuum limit of a lattice theory is very non-trivial. Most of the work in developing new algorithms in LFT has some connection to improve this point. In this thesis, we will also discuss some development in this direction using ML-based approaches.

2.2 Fermionic Field on Lattice:

Including fermions in LFT is not a straightforward task as it comes with a fermion doubling problem, where extra d.o.f appears in the theory. Even if we remove the doubling problem

with some existing methods the fermionic part in the lattice action makes simulation cost very high. In this thesis, we used the Gross-Neveu model, which incorporates fermions in the lattice action. I will give a quick overview of fermion field discretization. The free fermion action on Euclidean spacetime in the continuum can be written as

$$S_F^{cont.} = \int dx \bar{\psi}(x)[\gamma_\mu^E \partial_\mu + m]\psi(x). \tag{2.13}$$

where, $\psi(x)$ are spinor field represents fermions and γ_μ^E denotes the Dirac matrices in euclidean space. The naive fermion action after discretization of Euclidean spacetime

$$S_F = a^4 \sum_{n,m} \bar{\psi}(n) D_{nm} \psi(m). \tag{2.14}$$

where

$$D_{nm} = \sum_\mu \frac{1}{2}\gamma_\mu^E [\delta_{m,n+\hat{\mu}} - \delta_{m,n-\hat{\mu}}] + m\delta_{mn}.$$

Note that, ψ,$\bar{\psi}$ are Grassman variable.

2.2.1 Fermion Doubling:

In the momentum space, the Dirac propagator is

$$D(p) = \frac{1}{a}\sum_{\mu=1}^{4} \gamma_\mu^E \sin(ap_\mu). \tag{2.15}$$

And its inverse,

$$D^{-1}(p) = \frac{-ia}{\sum_{\mu=1}^{4} \gamma_\mu^E \sin(ap_\mu)}. \tag{2.16}$$

It has a pole at sixteen different locations- $\Big[(0,0,0,0), (\frac{\pi}{a},0,0,0), ..., (\frac{\pi}{a},\frac{\pi}{a},\frac{\pi}{a},\frac{\pi}{a})\Big]$. Now, if we take the continuum limit we will have 16 mass degenerate fermions from the poles at the corners of the Brillouin zone where 15 are unwanted or spurious d.o.f. For each dimension we get 2 degenerate fermions, hence a total of 16 in 4D spacetime. This problem is known as the fermion doubling problem. These fermions doublers in the lattice theory now can couple with gauge fields and produce quantum corrections. These doublers are responsible for the unphysical cancellation of the axial anomaly on the lattice. So, we need to remove

these doublers before doing any lattice simulations. We will discuss two popular methods which remove the doubler problem.

Wilson Fermions and Staggered Fermions: K.G. Wilson [5] proposed an approach where he adds a second derivative term in the action which vanishes in the continuum limit. Using this term one can remove the unwanted d.o.f but at the same time, it breaks the chiral symmetry explicitly. The action become

$$S_F^W = S_F + \frac{ar}{2}\bar{\psi}\Box\psi. \tag{2.17}$$

where r is the Wilson parameter and $\Box$ is the Euclidean Laplacian-

$$\Box = \sum_{\mu=1}^{4} \partial_\mu \partial_\mu.$$

The Dirac operator in momentum space is

$$D_W = m\mathbf{I} + \frac{1}{a}\sum_{\mu=1}^{4} \gamma_\mu \sin(ap_\mu) + \mathbf{I}\frac{r}{a}\sum_{\mu=1}^{4}(1 - \cos(ap_\mu)). \tag{2.18}$$

Now, we have one pole in Dirac propagator at $p_\mu = (0, 0, 0, 0)$ and another pole has effective mass $m_{eff} = 2n\frac{r}{a}$ and in the continuum limit $a \to 0$, the doubler mass $\to \infty$. In the end, we get only a single physical d.o.f in the continuum limit sacrificing the chiral limit.

There are alternative approaches which can remove the fermion doubling problem, and one of them, developed by Kogut and Susskind, is known as staggered fermions. The main idea behind staggered fermions is to distribute the components of the Dirac field, ψ, on different lattice points, resulting in a reduction of the degrees of freedom. Instead of having a single four-component Dirac field, the fermion field is split into four "tastes" or "flavours," represented by different lattice sites. Each taste corresponds to a subset of lattice points. As a result of this splitting, the number of fermions is reduced from 16 to 4. This reduction in degrees of freedom significantly reduces the computational complexity of lattice QCD simulations. In the massless limit, staggered fermions possess a remnant of chiral symmetry known as a chiral $U(1) \otimes U(1)$ symmetry. This symmetry is approximate and is subject to lattice artefacts, but it still allows for the study of chiral properties in lattice QCD calculations. The presence of this remnant chiral symmetry can be advantageous for studying phenomena related to chiral symmetry breaking and the properties of light quarks. Apart from these, there are several lattice fermion formalisms to reduce error and/or tame the fermion doubling problem such as Domain wall [75–77],

Overlap fermion [78, 79], Twisted mass fermion [80], Karsten-Wilczek [81], Borici-Creutz fermions [82–85] etc.

2.3 Lattice Gauge Theory

Lattice Gauge Theory deals with Quantum Field Theories which are invariant under the gauge transformations like Yang-Mills theory, QCD etc. Our work in this thesis is also motivated by lattice QCD which is a $SU(3)$ gauge theory. In this section, a concise introduction is provided on the formulation of Lattice Gauge Theory using a gauge symmetry of $SU(N)$.

In the previous section, we have seen that fermion sits on the lattice points and in the lattice action we have terms like $\bar{\psi}(n)\psi(n+\hat{\mu})$. Now, this term is not a gauge invariant term. Therefore, we have to introduce the concept of parallel gauge transporter from the continuum Gauge theory. In the continuum, it is defined as

$$G(x,y) = \mathcal{P}[e^{\int_c igAds}]. \tag{2.19}$$

where $\mathcal{P}$ is the path ordering operator, A is the gauge field in continuum theory, c is the path connecting two point x and y. In LFT we introduce an object called a link variable connecting two lattice point n and $n+\hat{\mu}$ in order to make the term $\bar{\psi}(n)\psi(n+\hat{\mu})$ gauge invariant. The link variable is defined as

$$U(n,n+\hat{\mu}) = U_\mu(n) = e^{ig\int_n^{n+a\hat{\mu}} A_\mu(x)dx^\mu} \equiv e^{igaA_\mu(n)} \in SU(N). \tag{2.20}$$

where,

$$A_\mu = \sum_{a=1} A_\mu^a \frac{\tau^a}{2}.$$

For $SU(3)$ gauge symmetry τ^a are the Gellmann matrices satisfying the following algebra

$$[\lambda_a, \lambda_b] = if_{abc}\lambda_c.$$

In lattice formulation, the lie algebra valued A_μ is not fundamental rather the group-valued link variable $U_\mu(n)$ are fundamental variables. The transformation of a link variable under $SU(N)$ group

$$U_\mu(n) \longrightarrow \Omega(n)U_\mu(n)\Omega_(n+\hat{\mu}). \tag{2.21}$$

Now, in order to construct a gauge invariant theory or action we can take the product of the link variable in a close loop (Wilson loop) C and take of the product.

$$W[U] = tr\Big[\prod_{(n,\mu)\in C} U_\mu(n)\Big]. \tag{2.22}$$

We can see under the $SU(N)$ gauge transformation, it is invariant

$$W[U] = tr\Big[\Omega(n_o)\prod_{(n,\mu)\in C} u_\mu(n)\Omega^\dagger(n_o)], \tag{2.23}$$

$$= tr\Big[\prod_{(n,\mu)\in C} U_\mu(n)\Big] = W[U]. \tag{2.24}$$

For a pure $SU(N)$ gauge action we can consider the smallest closed loop also known as

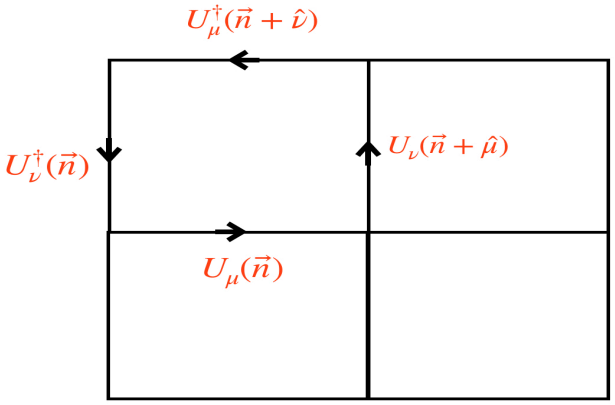


FIGURE 2.1: Plaqutte in terms of link variables

the plaquette as shown in Figure 2.1.

$$P_{\mu\nu}(U) = U_\mu(n)U_\nu(n+\hat{\mu})U_{-\mu}(n+\hat{\mu}+\hat{\nu})U_{-\nu}(n+\hat{\nu}), \tag{2.25}$$

$$= U_\mu(n)U_\nu(n+\hat{\mu})U_\mu^\dagger(n+\hat{\nu})U_\nu^\dagger(n). \tag{2.26}$$

And the gauge invariant action becomes

$$S_G[U] = -\sum_{\mu<\nu}\frac{\beta}{N}Re\Big(tr[P_{\mu\nu}(U)]\Big). \tag{2.27}$$

Now, the functional integral or the expectation of observable $O[U]$ can be computed as

$$< O >= \frac{1}{Z}\int\prod_{(n,\mu)} dU_\mu(n)O[U]e^{-S_G[U]}. \tag{2.28}$$

The path integral should be invariant under a change of variable e.g. under the $SU(N)$ gauge transformation.

$$Z = \int D[U'] \exp(-S_G[U']), \tag{2.29}$$

$$= \int D[U'] \exp(-S_G[U]). \tag{2.30}$$

The integration measure must satisfy $D[U'] = D[U]$. This measure is well defined and for details, one can look into *Haar measure* which is a measure for integration over a continuous compact group.

For $U(1)$ gauge theory, the link variables are not matrices so the U(1) action can be written as

$$S_G[U] = -\beta \sum_{\mu<\nu} Re\Big(P_{\mu\nu}(U)\Big), \tag{2.31}$$

where,

$$U_{n,n+\hat{\mu}} = e^{i\theta_\mu(n)}.$$

and the θ is restricted in a compact domain, $\theta \in [0, 2\pi]$.

2.4 Markov Chain Monte Carlo(MCMC)

MCMC simulation is an effective computational technique employed to approximate intricate probability distributions and examine systems characterized by high-dimensional parameter spaces. This method combines elements from Markov chain theory and Monte Carlo sampling to generate representative samples originating from the target distribution of interest.

The fundamental concept underlying MCMC simulation involves constructing a Markov chain in a manner that its stationary distribution coincides with the desired target distribution for sampling. In simpler terms, if the Markov chain is allowed to run for a sufficiently long duration, the sequence of samples it produces will eventually converge to the intended distribution.

To accomplish this, MCMC simulation relies on the principle of detailed balance or reversibility. This principle guarantees that the transitions between different states in the Markov chain occur based on a particular transition probability, deliberately designed to

satisfy the desired stationary distribution. By iteratively updating the current state of the chain using information from the preceding state, the chain explores the parameter space, gradually investigating regions with higher probabilities.

Let π be the target distribution and we want to generate a Markov chain in the state space X such that the stationary distribution of the chain is π. If at time t the chain is at x and the transition kernel is $T(x|y)$ then the chain will be stationary distribution if the following conditions are satisfied:

1. Erogodicity: Starting from any state in the Markov chain it must be possible to move to any other state in a finite number of steps.

2. Detailed balance:

$$p(y)T(x|y) = p(x)T(y|x). \tag{2.32}$$

Among the MCMC algorithms widely employed, notable examples include the Metropolis-Hastings (MH), Gibbs sampling, and Hybrid Monte Carlo (HMC). In this discussion, we will focus on providing a detailed overview of the MH and HMC algorithms, as these specific methods have been utilized for the thesis work.

2.4.1 Metropolis-Hasting

Metropolis–Hastings algorithm is the workhorse of MCMC simulation, because of its simplicity and its versatility and therefore the first solution to consider the intractable situations.

Algorithm 1 Metropolis-Hasting (MH); Target distribution: $\pi(x)$

- Initialize the Markov chain in the state space X; $X(t) = x(t)$, where $x(t)$ is the current state in Markov chain.
- Generate a proposal $Y(t) \sim q(y|x(t))$, where $q(y|x(t))$ is the proposal density conditioned on the current state
- Compute the quantity, $A = min\Big(1, \frac{\pi(y)q(x(t)|y)}{\pi(x(t))q(y|x(t))}\Big)$.
- Accept the proposal for the next step in the Markov chain,

$$X(t+1) = \begin{cases} Y(t), \text{ with probability } A(x(t)). \\ x(t), \text{ with probability } 1 - A(x(t)). \end{cases}$$

The Metropolis–Hastings algorithm while sampling from the target density π requires a choice of a conditional density $q(y|x)$ also known as a proposal or candidate kernel. let at time t, Markov chain is at $x(t)$, then the next sample y is proposed from a proposal distribution $q(y|x)$. The new proposal will be accepted in the Markov chain at $t+1$ with the probability,

$$A(y, x) = min\Big(1, \frac{\pi(y)q(x|y)}{\pi(x)q(y|x)}\Big). \tag{2.33}$$

If the proposal is not accepted then the chain stays in the current state.

Note that, the proposal density plays a crucial role in the efficiency and effectiveness of the algorithm. The choice of the proposal distribution should strike a balance between exploring the parameter space and exploiting regions of higher probability. By iteratively applying the Metropolis-Hastings algorithm and appropriately selecting the proposal distribution, we can generate a Markov chain that eventually converges to the desired target density π, allowing us to obtain representative samples from the target distribution.

2.4.2 The Metropolis Algorithm

This is a special case of the MH algorithm considering the proposal distribution as symmetric which was first proposed by Metropolis et al, 1953. If the proposal distribution is

symmetric i.e. $q(y|x) = q(x|y)$ then, A becomes

$$A(y, x) = \min\Big(1, \frac{\pi(y)}{\pi(x)}\Big). \tag{2.34}$$

Note that, the Metropolis algorithm can be slower then the Metropolis-Hasting algorithm. This is due to the fact that when the starting point in the Markov chain is far from the target distribution, the symmetric proposal distribution of the Metropolis algorithm may not be able to explore the parameter space efficiently. This can lead to slow convergence and a longer time to reach the target distribution.

2.4.3 Independent Metropolis-Hasting

The Independent or Generalized Metropolis-Hastings algorithm is a subclass of MCMC algorithms where the proposals are independent of the previous samples. This variant is also referred to as the Generalized MH algorithm. In the Independent MH algorithm, each proposal is generated independently without relying on the previous samples in the Markov chain. This characteristic can be advantageous in situations where the proposal distribution is already quite close to the target distribution. In such a situation, the independent nature of the proposals can facilitate quicker exploration and convergence towards the target distribution. The algorithm effectively exploits the favourable characteristics of the proposal distribution without the need for additional correlations introduced by previous samples. The acceptance probability becomes

$$A(y, x) = min\Big(1, \frac{\pi(y)q(x)}{\pi(x)q(y)}\Big). \tag{2.35}$$

It has been proven that the general MH algorithm satisfies both detailed balanced conditions and ergodicity, hence its stationary distribution is also the target distribution. We will be using this algorithm in Chapter 5 in the application of the Normalizing flow model for lattice simulation.

2.4.4 Hamiltonian Monte Carlo (HMC) Simulation

Hamiltonian Monte Carlo Simulation (HMC) (also known as Hybrid Monte Carlo) is the most popularly used MCMC sampling technique in the lattice community which can generate samples from a high dimensional probability distribution using the concept of Hamiltonian dynamics. The HMC simulation method is useful when the target distribution is

highly complex and has a correlation among the different dimensions. The basic idea of HMC is to simulate the dynamics of a Hamiltonian system in order to propose a new state to construct the Markov chain. The Hamiltonian system consist of a position variable and a momentum variable which evolve over time according to the laws of physics. The position variable corresponds to the parameter being sampled, while the momentum variable is introduced to help guide the dynamics towards the region of high probability. To perform the HMC simulation one has to define the Hamiltonian of the system which includes the information of the target distribution. If $\pi(x)$ is the target density then the Hamiltonian can be defined as

$$H(x,p) = -\log \pi(x) + \sum_{i=1}^{i=d} \frac{p_i^2}{2m_i}, \tag{2.36}$$

$$= U(x) + K(p). \tag{2.37}$$

where x is the d-dimensional position coordinate that we want to sample and p is the auxiliary momentum coordinate introduced to evaluate the Hamiltonian dynamics. The joint distribution of x and p can be defined in terms of the Hamiltonian as

$$p_{joint}(x,p) = \frac{1}{Z} \exp\Big(-\frac{H(x,p)}{T}\Big), \tag{2.38}$$

Where Z is the partition function of the system $Z = \sum \exp\left(-H(x,p)\right)$and T is the temperature of the system analogue to a statistical system. The auxiliary variable can be integrated out and we get the desired samples of x. The momentum variable p can be drawn from the distribution

$$P(p) \propto e^{-\frac{1}{2}pM^{-1}p} \approx \mathcal{N}(0, M), \tag{2.39}$$

$$\text{where, } M = \text{diag}(m_1, m_2 ... m_d). \tag{2.40}$$

So, we generate the auxiliary momentum from a multivariate independent Gaussian distribution from which sampling can be done easily.

Then, we have to solve Hamiltonian dynamics for the position and momentum variable, which are deterministic first-order differential equations of motion when the Hamiltonian is given. If $x(0)$ and $p(0)$ are given then we can solve for the trajectory of the system for

any time t, $\{x(t),\ p(t)\}\}$ as

$$\dot{x}_i = \frac{\partial H(x,p)}{\partial p_i}. \tag{2.41}$$

$$\dot{p}_i = -\frac{\partial H(x,w)}{\partial x_i}. \tag{2.42}$$

Let the initial point in phase space as $\{x(0), p(0)\}$ then applying the Hamiltonian dynamics for time τ the point will move to a new point in phase space $\{x(\tau), p(\tau)\}$. In the numerical simulation, we have to discretize the time and also solve Hamilton's equation of motion for a small finite step size. The most popular integration scheme used for HMC is the Leapfrog method. If the discretization step size is ϵ and N is the number of updates of dynamics before making a proposal then $\epsilon.N = \tau$. The Leapfrog update of both position and momentum is done in an alternate fashion. First, we do a half step for the momentum, then using the new values of the momentum we do a full step for the position, and again using the new values of the position variables, we do another half step for the momentum.

$$p_i(t+\epsilon/2) = p_i - (\epsilon/2)\nabla \log x(t). \tag{2.43}$$

$$x_i(t+\epsilon) = x_i(t) + \epsilon\frac{p(t+\epsilon/2)}{m_i}. \tag{2.44}$$

$$p_i(t+\epsilon) = p_i(t+\epsilon/2) - (\epsilon/2)\nabla \log(x(t+\epsilon)). \tag{2.45}$$

After such N step, we will generate a new phase point which will be the new proposal for the MH accept/reject steps. No matter whether we accept or reject the new point we will draw a new momentum from the Gaussian distribution and again run the dynamics which we call as Molecular Dynamics (MD) step.

The most important fact about generating a proposal using HMC is that even if the initial momentum vector points towards a lower-density region the Hamiltonian dynamics will change the momentum direction in such a way that it reaches the high probability region in the position space. So, the HMC generates better proposals using the information on target density in the potential function of the Hamiltonian.

In HMC the acceptance probability is also very high due to the conservation nature of Hamiltonian dynamics. Energy conservation can be easily shown in these dynamics.

$$\frac{dH(x(t),p(t))}{dt} = 0. \tag{2.46}$$

The acceptance rate in HMC is given by-

$$A = \min\left(1, \frac{\exp(-H(x(t+1), p(t+1)))}{\exp(-H(x(t), p(t)))}\right). \tag{2.47}$$

Since energy is conserved in the dynamics the from the above formula acceptance rate should be 100%. However, in numerical integration, we approximate the dynamics with step size which leads to dissipated Hamiltonian dynamics. Hence, in practice, the acceptance rate depends on the step size and number of steps used in the Molecular dynamics parts. The practical use of HMC simulation requires tuning two parameters of molecular dynamics, the step size ϵ and the number of MD steps N.

The acceptance rate depends on the step size, a large step size can lead to a low acceptance rate and too small a step size can lead to a longer run time in making a single proposal. Again, we can not use a low number of MD steps as it will lead to correlated samples which can lead to biased estimation. Therefore, we have to be careful in choosing the right values for these two parameters.

Finally, we need the theoretical guarantee that in the Markov chain, we are generating samples from the target density. It can be easily shown that the target density obeys the detailed balance condition under HMC.

$$\pi(y)T(x|y) = \pi(x)T(y|x). \tag{2.48}$$

The different steps one follows for practical implementation of HMC are shown in Algorithm 2.

Algorithm 2 Hamiltonian Monte Carlo (HMC)

- Draw the initial momentum from a Gaussian distribution, $p_{start} \sim \mathbf{N}(0, M^{-1})$.
- Define the initial position as x_0.
- Update momentum for half step-size:

$$p_0 = p_{start} - (\epsilon/2)\nabla \log p(x_0). \tag{2.49}$$

- Update position and momentum for j=1,2....N-1.

$$x_j = x_{j-1} + \epsilon.p_{j-1}, \tag{2.50}$$
$$p_j = p_{j-1} - \epsilon\nabla \log p(x_j). \tag{2.51}$$

- Final update on position:

$$x_N = x_{N-1} + \epsilon.p_{N-1}. \tag{2.52}$$

- Final update on momentum:

$$p_N = p_{N-1} - (\epsilon/2)\nabla \log p(x_N). \tag{2.53}$$

- Metropolis accept/reject step: Calculate acceptance probability:

$$A = \min\Big(1, \frac{\exp(-H(x_N, p_N))}{\exp(-H(x_0, p_0))}\Big). \tag{2.54}$$

- Accept x_N with probability A and set $x_{new} = x_N$ otherwise, if we reject then set $x_{new} = x_0$

Chapter 3

Introduction to Generative Model

Machine learning (ML) is an increasingly prominent subset of artificial intelligence that has gained significant attention in recent years due to its ability to develop algorithms and models that learn from data and utilize that knowledge for making predictions or decisions [86]. ML achieves this by employing mathematical and statistical techniques to uncover patterns and relationships within data, thereby enabling effective prediction and decision-making capabilities.

ML offers many exciting opportunities for research and development across diverse fields. Its potential to revolutionize industries and unlock previously unattainable applications is remarkable. A prime example is found within the healthcare sector, where machine learning algorithms can analyze patient data to predict their health outcomes [87]. Through ML, doctors can identify individuals at a heightened risk of developing specific diseases [88], empowering them to implement preventive measures before the conditions worsen. Similarly, financial institutions utilize machine learning algorithms to analyze market data, facilitating predictions regarding stock prices, interest rates, and other financial variables [89, 90].

But ML is not confined to industrial applications, it also has the potential to revolutionize scientific and engineering research. ML has found many applications in physics, ranging from data analysis to the discovery of new physical phenomena. ML is used in astrophysics to analyze large data sets from telescopes and observatories, such as the Hubble Space Telescope [91]. It is used to identify and classify galaxies, stars, and other celestial objects and to detect and predict astronomical phenomena. In particle physics, machine learning is used to analyze data from LHC [42–46]. ML is also used in the field of quantum

computing to optimize quantum circuits and algorithms, and to develop new quantum machine learning algorithms[92–96]. In LFT machine learning has been used for performing various task such as acceleration of simulation algorithm like HMC [49, 97–99] or development of new ML-based sampling method [60–63, 65] as well as for data analysis. As technology continues to advance, it is likely that we will see even more applications of machine learning in the future. Machine learning can be broadly categorized into three main categories:

Supervised Learning: Supervised learning is a type of machine learning algorithm where the model is trained using labelled data. Supervised learning aims to learn a mapping between input data and output labels based on a given set of examples[86]. The labelled data consists of input-output pairs, where the input x is the data to be analyzed, and the output y is the label or category assigned to that data. The model is trained by minimizing the difference between the predicted output $\hat{y}$ and the actual output y for each input-output pair or minimizing a loss function $\mathcal{L}$.

$$\theta = \text{argmin}(\mathcal{L}(y(x), \hat{y}(x, \theta))). \tag{3.1}$$

Examples of supervised learning algorithms include linear regression, logistic regression[100], decision trees, random forests, and neural networks.

Unsupervised Learning: Unsupervised learning is a type of machine learning where the algorithm learns patterns and relationships from input data without any explicit supervision or labelled output data. In unsupervised learning, the algorithm tries to identify hidden structures or patterns within the data, without any prior knowledge of what to look for. The goal of unsupervised learning is to discover interesting patterns and structures in data, such as clustering similar data points together, identifying outliers or anomalies, or dimensionality reduction, among others. Some popular unsupervised learning techniques include clustering algorithms like K-means[101], hierarchical clustering, and density-based clustering, as well as dimensionality reduction techniques like principal component analysis (PCA) and t-distributed stochastic neighbour embedding (t-SNE). Self-supervised learning is also an unsupervised learning method. Instead of relying on labelled data, self-supervised learning leverages the inherent structure or information present in the data itself to create supervisory signals for training. Word2Vec is a popular self-supervised learning algorithm used for learning word embedding, which represents words as dense vector representations. Skip-Gram is a popular algorithm used in word embedding models for self-supervised learning. It is designed to learn distributed representations of words by predicting the context of words given a target word.

Reinforcement Learning: In reinforcement leering machine learns through exploration and exploitation. In this type of learning, the machine is trained to make decisions in an environment based on feedback in the form of rewards or penalties. The machine learns through trial and error, with the goal of maximizing the cumulative reward over time [102]. Examples of reinforcement learning algorithms include Q-learning, policy gradients, and actor-critic methods.

There are also other subcategories of machine learning, such as semi-supervised learning: a combination of supervised and unsupervised learning, transfer learning: using pre-trained models for new tasks, and deep learning: using neural networks with many layers to learn complex representations.

In this chapter, we will give a detailed discussion about the concept and theory of generative models[103] and its various kinds which have been developed over the past few years. Generative models are semi-supervised ML models which can learn a given data distribution and able generate brand-new data. The goal of a generative model is to learn the underlying distribution of the input data so that it can generate realistic samples. It has many potential applications, including in the fields of art, music, and design. It can be applied to image synthesis, text generation, speech synthesis, music generation etc. For example, generative models have been used to create new pieces of art that are similar in style to the works of famous artists, such as Van Gogh or Picasso. In music, generative models have been used to generate new songs or pieces of music that are similar in style to a given set of training data. In text generation, a generative model can learn to generate new sentences or paragraphs that are similar in style and content to a given set of text.

In the context of LFT, we are interested in generative models which can learn a probability distribution defined by the lattice action and generate lattice samples from the distribution. As we have discussed earlier the dimensionality of a probability distribution in LFT is large which is equal to the total number of lattice points and hence the searches for suitable algorithms for efficient sampling are going on. We are looking for ML-based generating models which can sample such high dimensional lattice data efficiently. A range of generative models exists, including Variational Autoencoders (VAE), Boltzmann Machines (BM), Generative Adversarial Networks (GAN), Normalizing Flow (NF), and more, each exhibiting its own set of advantages and limitations. Extensive discussions have been dedicated to the GAN and Normalizing Flow models, which have been extensively employed in our research work. Additional information on other generative models can be found in the Appendix A.

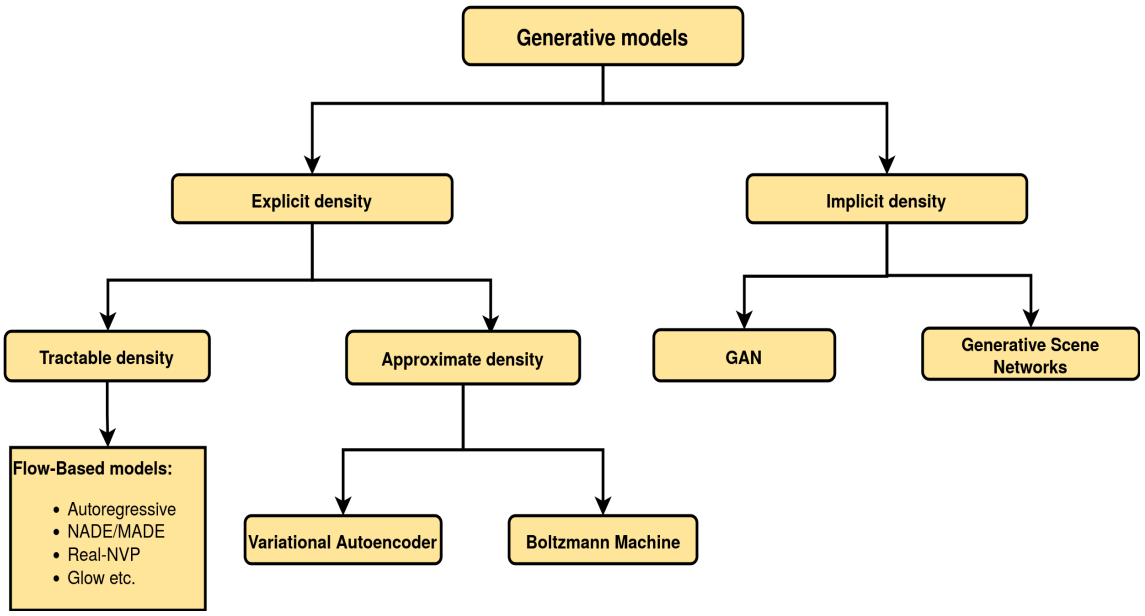


FIGURE 3.1: **Different Generative models**

3.0.1 Generative Adversarial Networks(GAN)

The concept of GAN was first introduced in a 2014 paper by Ian Goodfellow and his colleagues at the University of Montreal [104]. Goodfellow drew inspiration from a game-like scenario where two players, one generating realistic data and the other distinguishing it from real data, compete against each other. Since their introduction, GANs have found applications in diverse fields, ranging from image and video synthesis to text generation and even the creation of 3D models. They have also been employed for practical purposes, including data augmentation and enhancing the performance of other machine-learning models.

The GAN comprises two neural networks that engage in a competitive learning process. One of these networks, known as the generator, produces new data that resembles the dataset it was trained on. Conversely, the other network, referred to as the discriminator, distinguishes between real data from the dataset and artificial data generated by the generator. Throughout the training process, the generator aims to generate data that can deceive the discriminator into perceiving it as real, while the discriminator strives to accurately distinguish between real and synthetic data. Through iterative training, both networks improve their abilities, resulting in the generation of high-quality synthetic data. The architecture of the generator and discriminator is rooted in the deep learning concept of Neural Networks. In the Appendix A, we have provided a comprehensive discussion on essential terms such as Neural Networks, Fully Connected Neural Networks, Convolutional

Neural Networks, and Activation Functions, which are crucial components in constructing the GAN architecture.

Next, we will explore the process through which GANs can sample from a probability distribution characterized by high dimensionality. The objective is to train the generator to generate samples that adhere to the target probability distribution. This training is achieved by defining a loss function that evaluates the extent to which the generator's samples align with the desired distribution. The most popular loss function for GAN is called the Maximum Mean Discrepancy (MMD) loss or Min-Max loss. This loss measures the distance between the mean embedding of the target distribution and the generated distribution. In the training process, the generator network minimizes the MMD loss while the discriminator is trained to maximize it. This results in a generator that produces samples that closely match the target distribution. Once the generator is trained, samples can be generated by simply feeding random noise vectors into the generator and obtaining the output samples. The resulting samples should follow the target probability distribution.

Notably, GAN learns from the samples from the true distribution, without using the true distribution explicitly. Likewise, it generates samples and does not explicitly tell the probability density. The Generative model G, parameterized by Θ is a map from a random noise $z \sim p_z(z)$ to $x \sim p_g(G(z, \Theta))$. The training dataset is from a true distribution $x \sim p_{real}(x)$. The discriminator $D(x, \Phi)$ predicts whether it is coming from p_g or p_{real}. After completion of the training, we expect p_g to be as close as possible to p_{real}. The training process is a two-player min-max game where the discriminator improves its ability to distinguish the generator's fake samples and the generator also improves its ability to produce more realistic samples to fool the discriminator as the training continues. In the training process, both the generator and discriminator's weights are updated in tandem.

The objective function of GAN is:

$$\min_{G} \max_{D} V(G, D) = E_{x \sim p_{real}(x)}[\log D(x, \Phi)] + E_{z \sim p_z(z)}[\log(1 - D(G(z, \Theta), \Phi))]. \tag{3.2}$$

Now, the connection between the global optimum condition from the loss function and its relation to Kullback-Leibler (KL) divergence and Jensen-Shannon (JS) divergence will be explored. let $p_g(x)$ be the generator's distribution we are trying to train. For fixed G, the optimal value of D can be written as

$$D^*_G(x) = \frac{p_{real}(x)}{p_{real}(x) + p_g(x)}. \tag{3.3}$$

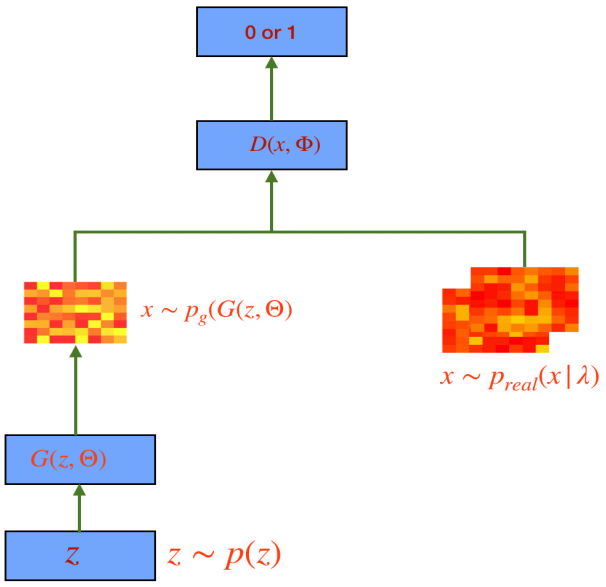


FIGURE 3.2: **Generative Adversarial Network**

The loss function can be re-write as

$$C(G) = max_D V(G, D), \tag{3.4}$$

$$= E_{x\sim p_{real}(x)}[\log D^*(x, \Phi)] + E_{z\sim p_z(z)}[log(1 - D^*(G(z, \Theta), \Phi))], \tag{3.5}$$

$$= E_{x\sim p_{real}(x}[\log D^*(x, \Phi)] + E_{z\sim p_g(x)}[log(1 - D^*(x, \Phi))], \tag{3.6}$$

$$E_{x\sim p_{real}}\Big[\log \frac{p_{real}(x)}{0.5(p_{real}(x) + p_g(x))}\Big], \tag{3.7}$$

$$+ E_{z\sim p_g}\Big[\frac{p_{real}(x)}{0.5(p_{real}(x) + p_g(x))}\Big]. \tag{3.8}$$

The Kullback–Leibler (KL) divergence between two probability distributions $p(x)$ and $q(x)$ is given by

$$KL(p||q) = \int p(x) \log \frac{p(x)}{q(x)dx}, \tag{3.9}$$

Similarly, the Jensen–Shannon (JS) divergence is given by

$$JS(p||q) = \frac{1}{2}KL\big(q||\frac{p+q}{2}\big) + \frac{1}{2}KL\big(p||\frac{p+q}{2}\big). \tag{3.10}$$

Now, $C(G)$ becomes

$$C(G) = KL\big(p_{data}||\frac{p_{data}+p_g}{2}\big) + KL\big(p_g||\frac{p_{data}+p_g}{2}\big) - 2\log 2, \tag{3.11}$$

$$= 2JS(p_{data}||p_g) - 2\log 2. \tag{3.12}$$

We know that JS divergence is always non-negative and zero only when the distributions are equal. Here, we found that $C^*(G) = -2\log 2$ is the global minimum that leads to $p_g = p_{data}$. This implies that the generator becomes perfect in generating samples which look like the true sample. Therefore, the loss function of GAN has a relation to both KL and JS divergence. It is also notable that the loss Equation (3.2) may not give sufficient gradient so that G can learn well. In the early stage of training the discriminator may reject all the samples since the generator is too poor to produce a realistic sample. Then the term $\log(1 - D(G(z)))$ saturates. Therefore one should maximize $\log D(G(z))$ instead of minimizing $\log(1 - D(G(z)))$, which will also give the same global minimum. A GAN architecture for sampling lattice Gross-Neveu in 2D for a single action parameter is discussed in the Appendix A.

Although GAN generates very high-quality samples, GAN training is notoriously difficult and prone to various challenges that must be carefully addressed during training:

Mode collapse: GANs can encounter a phenomenon known as mode collapse, wherein the generator becomes fixated on producing a restricted range of outputs, disregarding the other potential outputs. This situation arises when the generator repeatedly generates samples from a specific mode of distribution, and the discriminator consistently identifies them as authentic. Consequently, the generator intensifies its focus on generating more samples solely from that particular mode. As a consequence, the output can become monotonous and limited in variation. Consequently, GANs face challenges when attempting to learn a multi-modal distribution under normal conditions.

Vanishing gradients: During GAN training, a minimax game is employed, wherein the generator and discriminator are trained to optimize opposing objectives. However, this approach can give rise to a phenomenon known as vanishing gradients, wherein the gradients used to update the model weights become excessively small. As a consequence, the learning process may become unstable.

Unstable training: The training of GANs can present difficulties due to the need for careful hyperparameter tuning, including factors such as the learning rate, batch size, and

regularization. Minor adjustments to these hyperparameters can result in unstable training dynamics, causing the generator and discriminator to struggle to converge effectively.

Computationally expensive: GANs can be computationally expensive to train, especially for high-resolution images or larger lattice sizes. This can be challenging for researchers or organizations with limited computational resources.

Addressing these challenges requires a deep understanding of GANs and their underlying mechanisms, as well as careful experimentation and tuning of hyperparameters and training strategies. Researchers continue to develop new techniques and approaches to overcome these challenges and improve the performance of GANs.

3.1 Normalizing Flow

Normalizing flow [105, 106] is a family of probabilistic models that possess the capability to transform a simple distribution, such as a Gaussian distribution, into a more complex distribution with desired properties. The concept of normalizing flow was initially introduced by Dinh et al. in 2014. In their work, the authors proposed a bijective transformation achieved by composing a series of invertible and differentiable functions, each progressively transforming the input distribution into a more intricate form. Notably, these transformations can be efficiently inverted, allowing for effective inference and sampling. Since its introduction, normalizing flow has gained popularity as a generative modelling technique within the field of machine learning.

Normalizing flow offers an alternative approach to generative models by explicitly modelling the probability distribution of the data. The primary advantage of using normalizing flow models lies in their ability to effectively model complex and high-dimensional data distributions while maintaining a tractable likelihood function. This trait proves invaluable for a variety of applications, including density estimation, image generation, and anomaly detection. The term "normalizing flow" derives from the fact that these transformations can be conceptualized as a flow of probability density from a simple distribution to a more complex one, with each transformation "normalizing" the density along the way. Once trained on a specific dataset, the normalizing flow can be utilized to generate new samples that exhibit similarity to the original data, making it a powerful tool for tasks such as image and text generation, anomaly detection, and density estimation. Normalizing flow models have demonstrated exceptional performance on various benchmark datasets and remain an active area of research within the field of machine learning.

3.1.1 Theory of Normalizing Flow

Let us consider Z and Φ are two random variables $Z, \Phi \in R^D$, i.e. both are D dimensional random variables. Now consider a invertible and smooth mapping from z to ϕ as $\phi = g(z)$ with inverse $g^{-1} = f$. If $p_Z(z)$ and $q_\Phi(\phi)$ are the probability distribution of Z and Φ respectively, we can write their relation using the variable transformation formula as follows

$$q_\Phi(\phi) = p_Z(z)\left|\det J_g(z)\right|^{-1}, \tag{3.13}$$

$$= p_Z(g^{-1}(\phi))\left|\det J_{g^{-1}}(g^{-1}(\phi))\right|^{-1}. \tag{3.14}$$

Here, the Jacobian is defined as $D \times D$ matrix:

$$J_g(z) = \begin{bmatrix} \frac{\partial g_1}{\partial z_1} & \cdots & \frac{\partial g_1}{\partial z_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_D}{\partial z_1} & \cdots & \frac{\partial g_D}{\partial z_D} \end{bmatrix}. \tag{3.15}$$

If we assume that the prior distribution $p_Z(z)$ is a simpler one which has a tractable

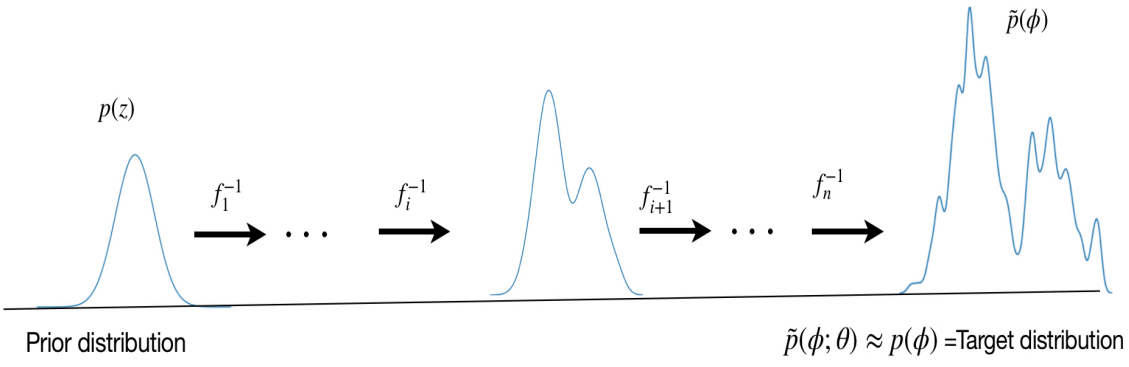


FIGURE 3.3: Normalizing flow: We start with a simple distribution which gets transformed into a more complicated distribution by applying a series of functions g_i or f_i^{-1}.

density and sampling can be done easily then using the above relation we can perform the following two tasks:

Sampling: To generate a sample in Φ one can sample z from the prior distribution $z \sim p_Z(z)$ and apply the forward function $\phi = g(z)$. In this case, we do not need to know the inverse of the forward function.

Density Estimation: For estimating density for a sample in Φ space we require the inverse of the forward function i.e. g^{-1} should be known and its jacobian must be computable. Also, the prior distribution must be tractable i.e. we must know how to calculate the density $p_Z(z)$.

The function g can be arbitrarily complicated to represent a complicated distribution on Φ. However, as we have seen g must be invertible and its determinant of jacobian should be computable. We can construct such a complicated function using the idea that the composition of an invertible function is also invertible and its jacobian has a specific form. Let $g_1, g_2, g_3 ... g_N$ are N bijective functions then their composition can be defined as $g = g_N \circ g_{N-1} \circ ... g_1$. Then g is also bijective and its inverse function can be written as

$$\begin{aligned} g^{-1} = (g_N \circ g_{N-1} \circ ... g_2 \circ g_1)^{-1} &= g_N^{-1} \circ g_{N-1}^{-1} \circ ... g_2^{-1} \circ g_1^{-1}, \\ &= f_N \circ f_{N-1} \circ ... f_2 \circ f_1. \end{aligned} \tag{3.16}$$

The determinant Jacobian is given as

$$DetJ_{g_N \circ ... g_2 \circ g_1} = DetJ_{g_N}(g_{N-1}(z')) ... DetJ_{g_2}(g_1(z)).DetJ_{g_1}(z), \tag{3.17}$$

where $z' = (g_{N-2} \circ g_{N-3} \circ ... g_2 \circ g_1)(z)$.
Normalizing flow can model a complicated distribution but can they represent any distribution even if we take a base distribution which is very simpler? One can show that for any well-behaved target distribution $p^T(\phi)$ and prior distribution $p_Z(z)$ there exists a diffeomorphism that can transform $p_Z(z)$ to $p^T(\phi)$[107]. We have seen that we can compose many invertible functions to map z to ϕ i.e. $\phi = g(z)$. where, $g = g_N \circ g_{N-1} \circ ... g_2 \circ g_1$. Now, we can use g_i as a Neural network with learnable parameters. The model density can be parameterized by Neural Network weight and biases i.e θ as

$$\begin{aligned} q_\Theta(\phi, \theta) &= p_Z(g(z;\theta)) \left| \det J_g(z;\theta) \right|^{-1}, \\ &= p_Z(g^{-1}(\phi;\theta)) \left| \det J_{g^{-1}}(g^{-1}(\phi;\theta)) \right|^{-1}, \end{aligned} \tag{3.18}$$

where, $\phi = g(z;\theta)$.

If the target distribution is $p_T(\phi)$ then we can optimize the network parameter such that the discrepancy between the target density and model density $q_\Theta(\phi, \theta)$ is minimized. This can be done by minimizing the KL divergence between the above-mentioned distribution. There are two different optimization schemes - (1) Forward KL divergence and (2) Backward or Reverse KL divergence, as described below which are used considering the problem at hand.

1. Forward KL divergence: Training a flow-based model by Forward KL divergence requires training samples from the target distribution. It is equivalent to maximum likelihood estimation. It can be defined as

$$\begin{aligned}\mathcal{L}(\theta) =& D_{KL}[p_T(\phi)||q_\Theta(\phi,\theta)],\\ =& - E_{p_T}[\log q_\Theta(\phi,\theta)] + const,\\ =& - E_{p_T}[\log p_z(g^{-1}(\phi,\theta))] + \log|detJ_{g^{-1}}(\phi,\theta)| + const.\end{aligned} \tag{3.19}$$

Now if we have $\{\phi_i\}_{i=1}^N$ samples from the target density $p_T(\phi)$ then we can estimate the expectation over $p_T(\phi)$ as follows

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\log p_z(g^{-1}(\phi_i,\theta)) + \log|detj_{g^{-1}}(\phi_i,\theta)| + const.$$

Minimizing the above loss function is equivalent to fitting the flow-based model by maximum likelihood estimating using the $\{\phi_i\}_{i=1}^N$ samples. The optimization of flow model parameters θ can be done iteratively with a stochastic gradient-based method

$$\nabla_\theta\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\nabla_\theta\log p_z(g^{-1}(\phi,\theta)) + \nabla_\theta\log|detJ_{g^{-1}}(\phi,\theta)|$$

. So, to use Forward KL divergence we need to compute g^{-1}, its Jacobian determinant and the prior density. Also, all three quantities must be differentiable w.r.t the parameters.

2. Reverse KL divergence There is a second kind of loss function which does not require samples from the target distribution known as Reverse KL divergence. The loss function can be written as

$$\begin{aligned}\mathcal{L}(\theta) =& D_{KL}\Big[q_\Theta(\phi,\theta)||p_T(\phi)\Big],\\ =& E_{q_\Theta}\Big[\log q_\phi(\phi,\theta) - \log p_T(\phi)\Big],\\ =& E_{p_z}\Big[\log p_z(z) - \log|detJ_{g(z)} - \log p_T(g(z,\theta))|\Big].\end{aligned} \tag{3.20}$$

Note that we can minimize $\mathcal{L}$ even if f $p_T(\phi)$ is known up to a normalization constant. That is we can write $p_T(\phi) = \frac{\tilde{p}_T(\phi)}{Z}$ where, Z is a normalization constant and $\tilde{p}_T(\phi)$ is a tractable density. Then the loss function becomes

$$\mathcal{L}(\theta) ==E_{p_z}\Big[\log p_z(z) - \log|detJ_{g(z)} - \log\tilde{p}_T(g(z,\theta))|\Big] + const. \tag{3.21}$$

Since samples from the prior can be easily sampled so, let us use $\{z\}_{i=1}^{N}$ samples from $p_z(z)$ and calculate the gradient of the loss function as follows

$$\nabla_\Theta(\mathcal{L}(\theta)) = -\frac{1}{N}\sum_{i=1}^{N}\nabla_\theta \log|detj_g(z,\theta)| + \nabla_\theta\tilde{p}_T(g(z,\phi)).$$

In order to train a flow-based model using reverse Kl divergence we need a sample from the prior distribution, we must compute g and its Jacobian determinant.

3.2 Construction of Flow model:

As we have discussed earlier, building a flow model requires designing the function g which has to be invertible, must be differentiable and the Jacobian determinant must be computable. Such constructions are nontrivial since the model must be expressive enough to properly learn a complicated distribution. In this section, we will discuss coupling flows including affine coupling layers, which are used in this thesis for the construction of the flow model. A short introduction to Auto-regressive flows is also presented in the Appendix A.

3.2.1 Coupling Layers

The coupling layer divides the input, z, into two segments and independently transforms the second segment based on the first. When combining multiple coupling layers, the elements of z are permuted among the layers to ensure that all dimensions undergo transformation. Specifically, in the first coupling layer, z is divided into two parts: $z = [z_{\leq d}, z_{\geq d}]$. The first part is transformed using a neural network without incorporating any information from the second part. Meanwhile, the second part is element-wise transformed in a

manner that relies on the first part. The transformation can be represented as follows:

$$z'_{\leq d} = z_{\leq d}, \tag{3.22}$$

$$(h_{d+1}, .., h_D) = \text{NN}(x_{\leq d}), \tag{3.23}$$

$$z' = \tau(z_i; h_i) \text{ for } i > d. \tag{3.24}$$

The inverse transformation can be written as

$$z_{\leq d} = z'_{\leq d}, \tag{3.25}$$

$$(h_{d+1}, .., h_D) = \text{NN}(x_{\leq d}), \tag{3.26}$$

$$z = \tau(z'_i; h_i) \text{ for } \ i > d. \tag{3.27}$$

The Jacobian matrix for the transformation is

$$J = \begin{bmatrix} I & 0 \\ R & \mathcal{P} \end{bmatrix}. \tag{3.28}$$

The dimensions of different matrices are

$0 \to d \times (D - d)$,
$I \to d \times d$,
$R \to (D - d) \times d$,
$\mathcal{P} \to (D - d) \times (D - d)$.

Coupling layers have emerged as a highly effective technique for implementing fast and efficient flow-based models, enabling both rapid sampling and density estimation. They have proven particularly successful in the realm of generative models for sampling high-dimensional data distributions, including images, audio, videos, and more. Notably, various well-known generative flow-based models, such as NICE, Real NVP, Glow, WaveGlow, and FlowWaveNet, have extensively utilized coupling layers.

There are different coupling function one can use which has been developed in the recent few years. One such popular coupling transformation is an affine transformation.

3.2.2 Affine Coupling

In affine coupling, the input is split into two parts: one part is transformed without modification, while the other part is transformed via an affine transformation. The affine

transformation consists of a linear transformation (matrix multiplication) and a translation (vector addition). The parameters of the affine transformation are learned during training, usually via maximum likelihood estimation. Affine coupling is invertible and its Jacobian is also computable.

First, we split the input into two parts:

$$z = [z_{\leq d}, z_{\geq d}]$$

let us say $z_{\leq d} = \mathbf{z}_1$ and $z_{\geq d} = \mathbf{z}_2$. Then we keep the second part $\mathbf{z}_2$ unchanged by the first affine coupling layer.

$$\phi_2 = \mathbf{z}_2. \tag{3.29}$$

However, we transform the second part using Neural Network to learn the parameters of

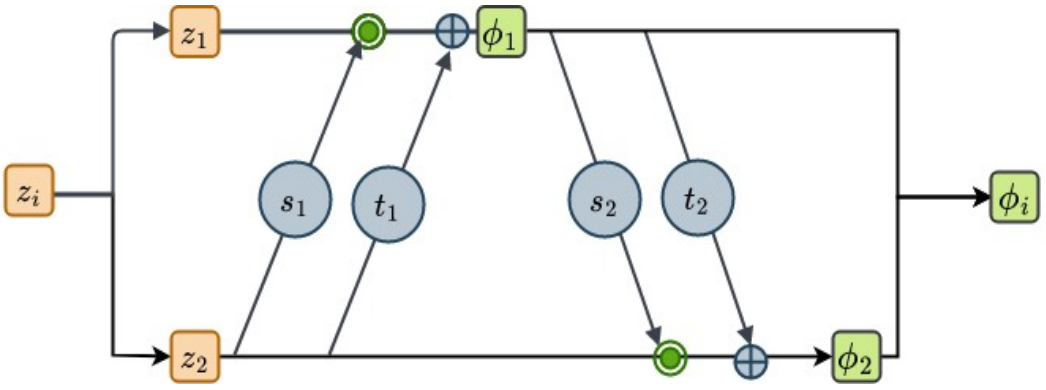


FIGURE 3.4: An affine coupling layer, where z is splitted into two parts (z_1, z_2) and s and t represent two neural networks.

the affine transformation for the transformation of the first part $\mathbf{z}_1$.

$$(\mathbf{s}, \mathbf{t}) = \mathrm{NN}(\mathbf{z}_2). \tag{3.30}$$

Now we transform the first part using the affine transformation-

$$\phi_1 = \mathbf{s} \odot \mathbf{z}_1 + \mathbf{t}. \tag{3.31}$$

In the next coupling layer, the role of $\mathbf{z}_1$ and $\mathbf{z}_2$ will reverse play by ϕ_2 and ϕ_1. From the Figure 3.4 we can see the $\mathbf{s}$ and $\mathbf{t}$ contain the neural network weight and biases and one does not need the inverse of this, as affine transformations are invertible even if one compute $\mathbf{s}$ and $\mathbf{t}$ only in the forward direction. We can use either CNN or a dense neural network

based on the problem. We can combine many such affine coupling layers to construct a flow model to make it more expressive and powerful.

Chapter 4

Application of GAN for the Simulation of Lattice Gross-Neveu Model

In this chapter, our focus will be on utilizing GANs to sample a specific probability distribution defined by the lattice Gross Neveu model [108] in (1+1) dimensions which is a field theory in Euclidean space. As discussed earlier, GANs have demonstrated their capability to generate high-quality images surpassing other generative models. Moreover, they have proven effective in sampling high-dimensional probability distributions across various domains. However, when it comes to lattice field theory, acquiring training samples from the lattice distribution poses challenges compared to obtaining natural images.

In the context of GANs, the learning process involves capturing the underlying probability distribution from the provided training samples. Therefore, when it comes to learning the lattice distribution using GAN, we would require pre-existing samples from the lattice distribution. However, in lattice field theory, there is no inherent advantage in terms of obtaining such training samples, as they are not readily available like natural images. However, our idea lies in learning a conditional distribution over the action parameter of the Gross Neveu model. The Gross-Neveu model has one action parameter or coupling λ, upon changing λ, the distribution also changes. Therefore, if we are able to learn a conditional distribution over the λ parameter, we can generate samples for a wide range of λ where generation is difficult using existing simulation methods. So, we skip the critical region in the training process and take training samples only from the non-critical region where the simulation cost is lower. After the completion of training, samples near the

critical region can be generated by specifying the parameter alone. It is important to note that the lattice size remains fixed throughout both the training and sampling processes. Therefore, it is advisable, to begin with a large lattice size during training to ensure accurate estimation of observables near the critical region.

The primary advantage of employing a GAN is a significant reduction in autocorrelation among the generated samples. GANs inherently produce uncorrelated samples, resulting in independent lattice samples with minimal autocorrelation in the vicinity of the critical region. Once the training phase is over, the lattice generation process within the GAN becomes very fast. Leveraging the learned conditional distribution, we can generate samples from the model for various action parameters, enabling parallel generation. Implementing conditional learning will require imposing a condition on the GAN. To accomplish this, we must construct a conditional GAN architecture appropriate for our problem.

4.0.1 Conditional-GAN:

To establish a foundation for our sampling method, it is important first to introduce the concept of the conditional Generative Adversarial Network (C-GAN). In standard GAN approaches, the generator's output is solely determined by the input random noise, lacking control over the specific type or class of data generated when dealing with datasets that contain categories or classes. However, there are instances where it becomes crucial to generate data of a particular type or class. In our problem, we want to label each lattice with its corresponding distribution $P(\phi|\lambda)$, i.e. the parameter λ, while generating from the generator. To address this requirement, we aim to train a GAN that can learn a conditional probability distribution, allowing for more control over the generated output.

In conditional-GAN [109] one appends the random noise with additional information λ, which could be attributes or class labels to produce output $G(\lambda, z, \Theta)$, which is conditioned on λ. We can append λ to the input of the discriminator as well.

In Chapter 3, we discussed a GAN's objective or loss function in detail. The loss function for conditional GAN modifies to

$$\min_{G}\max_{D} V(\lambda, G, D) = E_{x\sim p_{real}(x|\lambda)}[logD(\lambda, x, \Phi)] + E_{z\sim p_g(z)}[log(1 - D(\lambda, G(\lambda, z, \Theta), \Phi))]. \tag{4.1}$$

We will be using this loss function for training the C-GAN model.

4.1 Gross-Neveu Model in 1+1 Dimensions

This section will discuss the theory of the Gross-Neveu (GN) model on both continuum and discrete spacetime.

The Gross-Neveu model [108] is a simple quantum field theory that describes the behaviour of fermions interacting via a four-fermion interaction. The Gross-Neveu model is an example of a chiral symmetry breaking theory, where a symmetry between left-handed and right-handed fermions is broken. The model is often used as a simplified model for studying the behaviour of strong interactions in particle physics. The Gross-Neveu model has been studied in many different contexts in theoretical physics, including condensed matter physics, high-energy physics, and string theory. The Gross-Neveu model can be used for studying the behaviour of QCD in certain limits. For example, the Gross-Neveu model can be used to study the behaviour of QCD in the large N limit, where N is the number of colours in QCD. In this limit, the Gross-Neveu model and QCD exhibit similar behaviour, allowing physicists to gain insight into the behaviour of QCD in a simplified setting.

4.1.1 Continuum Theory

The Euclidean Lagrangian of GN model in 1+1 dimension is [108]:

$$\mathscr{L} = \sum_{f=1}^{N_f} \overline{\psi}_f(x)\partial\psi_f(x) - \frac{g^2}{2}\Big(\sum_{f=1}^{N_f} \overline{\psi}_f(x)\psi_f(x)\Big)^2. \tag{4.2}$$

With the help of the so-called Hubbard-Stratonovich(HS) transformation, we can reduce the four fermion part to a term quadratic in the fermion fields and an additional auxiliary bosonic field. The transformation is basically a shifted Gaussian integral.

$$\begin{aligned} &\exp(-\int d^2x[\frac{g^2}{2}(\overline{\psi}_f(x)\psi_f(x))^2], \\ &= \mathcal{N}[\int \mathscr{D}\sigma(x) exp(-\int d^2x[\tfrac{N_f}{2\tilde{\lambda}}\sigma^2(x) + \overline{\psi}_f(x)\sigma(x)\psi(x)]. \end{aligned} \tag{4.3}$$

where $\tilde{\lambda} = g^2 N_f$.

The partition function becomes

$$Z = \int \mathscr{D}\overline{\psi}\mathscr{D}\psi\mathscr{D}\sigma e^{-S_\sigma[\overline{\psi},\psi,\sigma]}. \tag{4.4}$$

The action is given by

$$S_\sigma[\overline{\psi},\psi,\sigma] = \int d^2x[\tfrac{N_f}{2\tilde{\lambda}}\sigma^2(x) + \overline{\psi}_f D_{GN}(x)\psi_f(x)], \tag{4.5}$$

where, $D_{GN} = \partial + \sigma(x)$. One can show that the σ field and condensate field $\overline{\psi}\psi$ are linked via,

$$\langle\overline{\psi}(x)\psi(x)\rangle = \frac{-N_f}{\tilde{\lambda}}\langle\sigma(x)\rangle. \tag{4.6}$$

Therefore, the average of the auxiliary field $\langle\sigma(x)\rangle$ can be referred to as the Chiral Condensate. This $\langle\sigma(x)\rangle$ can be used as an order parameter to study spontaneous chiral symmetry breaking of the GN model. GN model is analytically solvable in the infinite flavour limit: $N_f \to \infty$. Its phase structure has been studied extensively in this limit [110, 111]. Inhomogeneous phases of the GN model in the lattice are also studied for a finite number of flavours with proper continuum limit in [112, 113].

4.1.2 Lattice theory

The action of the lattice GN model in the staggered formalism [114] is generally written as

$$S = \sum_{x,y}[\frac{\lambda N_f}{2}\sigma^2(x) + \sum_{f=1}^{f=N_f}\overline{\chi}_f(x)D(x,y)\chi_f(y)]. \tag{4.7}$$

where the coupling constant is inverted to $\lambda = 1/\tilde{\lambda}$ for simulation purpose and $D = D_1 + \Sigma$ with

$$D_1(x,y) = \frac{1}{2}[\delta_{x,y+\hat{1}} - \delta_{x,y-\hat{1}}] + \frac{1}{2}[\delta_{x,y+\hat{2}} - \delta_{x,y-\hat{2}}]. \tag{4.8}$$

$$\Sigma_{xy} = \frac{1}{4}\delta_{xy}[\sigma(x) + \sigma(x-\hat{1}) + \sigma(x-\hat{2}) + \sigma(x-\hat{1}-\hat{2})]. \tag{4.9}$$

where $\hat{1}$ and $\hat{2}$ are unit vectors in the two directions in 2D.
This theory has discrete chiral symmetry

$$\chi \to (-1)^{x_1+x_2}\chi, \quad \overline{\chi} \to -(-1)^{x_1+x_2}\overline{\chi}, \quad \sigma(x) = -\sigma(x). \tag{4.10}$$

A higher N_f value is necessary to match continuum ($N_f \to \infty$) results, but $N_f = 2$ will serve our purpose for the discussions. After introducing pseudofermionic [69] method (for $N_f = 2$) action become non-local

$$S[\sigma,\phi,\lambda] = \sum_{x,y}[\frac{\lambda}{4}\sigma^2(x) + \phi^\dagger(x)(M^{-1})\phi(y)], \tag{4.11}$$

where $M = D^\dagger D$ and ϕ are pseudofermionic complex field.
The partition function can be written as

$$Z = \int \mathscr{D}\sigma \mathscr{D}\phi^\dagger \mathscr{D}\phi e^{-S[\phi,\phi^\dagger,\sigma]}. \tag{4.12}$$

With the action given in Equation (4.11), we perform the HMC simulations whenever required during numerical experiments. In this work, we have used the staggered fermion (for details about staggered fermion, see [68]) for lattice simulation.

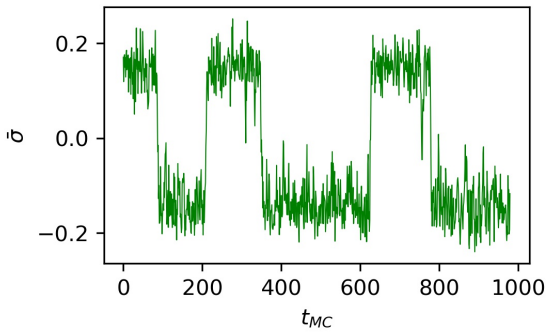


FIGURE 4.1: Fluctuations of $\bar{\sigma}$ values for lattice configurations during HMC simulation in between two minima at $\lambda \lessapprox \lambda_{crit}$.

In our numerical study, we use two observables on the lattice: $\langle|\bar{\sigma}|\rangle$ and χ for evaluation of C-GAN model. The quantity: $\bar{\sigma} = \frac{1}{N}\sum_x \sigma(x)$, which is measured in a single lattice configuration, can be used to study the phase transition as its ensemble average has a direct relation to Chiral Condensate $\langle\overline{\psi}\psi\rangle$. However, there is a problem with quantity $\bar{\sigma}$ as its ensemble average $\langle\bar{\sigma}\rangle$ vanishes even for, $\lambda \lessapprox \lambda_{crit}$ i.e. even for broken phase close to the critical point. So instead of using $\langle\bar{\sigma}\rangle$, we choose $\langle|\bar{\sigma}|\rangle$ as our order parameter which is a suitable observable to study phase transition. One more observable of importance is susceptibility. The two observables can be defined as

$$\langle|\bar{\sigma}|\rangle = \frac{1}{N}\langle|\sum_x \sigma(x)|\rangle; \quad \chi = N[\langle\bar{\sigma}^2\rangle - \langle|\bar{\sigma}|\rangle^2]. \tag{4.13}$$

where N is the lattice volume.

4.2 Proposed Method

We aim to train a conditional GAN (C-GAN) on lattice samples generated by MCMC simulations like HMC for different parameter values of λ. However, in the HMC simulation

of the lattice Gross-Neveu model, we sample both σ, ϕ field according to the distribution as defined in Equation (4.12)

$$P(\sigma, \phi|\lambda) = \frac{1}{Z} e^{-S(\sigma,\phi|\lambda)}, \tag{4.14}$$

where $S(\sigma, \phi)$ is the lattice action defined in Equation (4.11). We want the C-GAN to learn the marginal distribution of σ field, i.e. $p(\sigma|\lambda)$. So we discard HMC-generated pseudofermionic ϕ samples and only consider σ samples, which now represent the marginal distribution of σ from the joint distribution in Equation (4.14). Let the samples from the C-GAN represent an implicit distribution $\hat{p}(\sigma|\lambda)$. Our target is to train the C-GAN so that $\hat{p}(\sigma|\lambda)$ approximates the true distribution $p(\sigma|\lambda)$. To address the problem of critical slowing down, we train the C-GAN model for λ values sampled in non-critical regions, where the autocorrelation time is much smaller compared to the critical region. Hence, the generation of the training dataset is not affected by critical slowing down. Then we use the trained C-GAN model to generate samples near critical λ. Since the C-GAN model generates independent samples, hence our method can produce uncorrelated lattice configurations in the critical region.
Vanilla C-GAN trained over the HMC samples fails to learn the distribution reliably. The learning is made efficient as well as robust by incorporating into the C-GAN model the information of symmetries and constraints in the theory. Also, transforming the samples to reduce the imbalance in their values improves learning. We discuss these in detail in the following subsections.

4.2.1 Translation Symmetry

Due to translation symmetry in GN model lattices, the C-GAN generator made of dense layers fails to learn the true distribution properly. Convolutional kernels allow translational invariance in the lattices. Hence, using convolutional layers in the generator allows the learning to take place efficiently.

4.2.2 Transformation of sigma field

Note that, the observables of the GN model can be calculated from $|\bar{\sigma}|$, hence for training the C-GAN we transformed the lattice configurations such that each configuration has $\bar{\sigma} > 0$. This will reduce degrees of freedom for the C-GAN model, which will help in exploring the distribution space more efficiently. For that purpose, we select a particular

configuration and if found $\bar{\sigma} < 0$ then we apply a transformation:

$$\sigma(x) = -\sigma(x), \quad \forall x. \tag{4.15}$$

For training purposes of C-GAN, we apply natural log transformation to the HMC-generated samples as follows

$$\sigma'_i(x) = ln(\sigma_i(x) + c), \quad \forall \text{x,i} \tag{4.16}$$

where i represents a single lattice configuration from the ensemble and c is a constant such that the sum inside the logarithm become positive. This transformation (Equation (4.16)) become necessary for stable training of C-GAN as it balances data values and reduces the dynamical range of the $\sigma(x)$ field.

For efficient training, we apply the Min-Max scaling to the above-transformed data to bring it into a range [-1,1].

4.2.3 Periodic Boundary Condition

During the generation of configurations by HMC, we apply periodic boundary condition, i.e. we replace $\sigma(i, j)$ by $\sigma'(i, j) := \sigma((i)_N, (j)_N)$, where $(i)_N$ represents i modulo N. In order to learn the periodicity by the C-GAN model we apply periodic padding to all layers of the generator and initial two layers of the discriminator.

4.3 C-GAN Arcitecture and Training

4.3.1 C-GAN Architecture

Ensuring a stable training process requires the development of an optimistic architecture for the conditional GAN. In Appendix A, we discuss the architecture of a GAN. Now, we will present the architecture of the C-GAN, where the inclusion of conditional parameters becomes crucial. The model needs to effectively process both the conditional parameter λ and its corresponding lattice sample for accurate generation.

The input to the generator model consists of a 4×4 i.i.d Gaussian noise with zero mean and unit variance, stacked together with a 4×4 matrix containing all entries as λ. In the generator model, three 2D Transposed convolutional layers are used for up-sampling

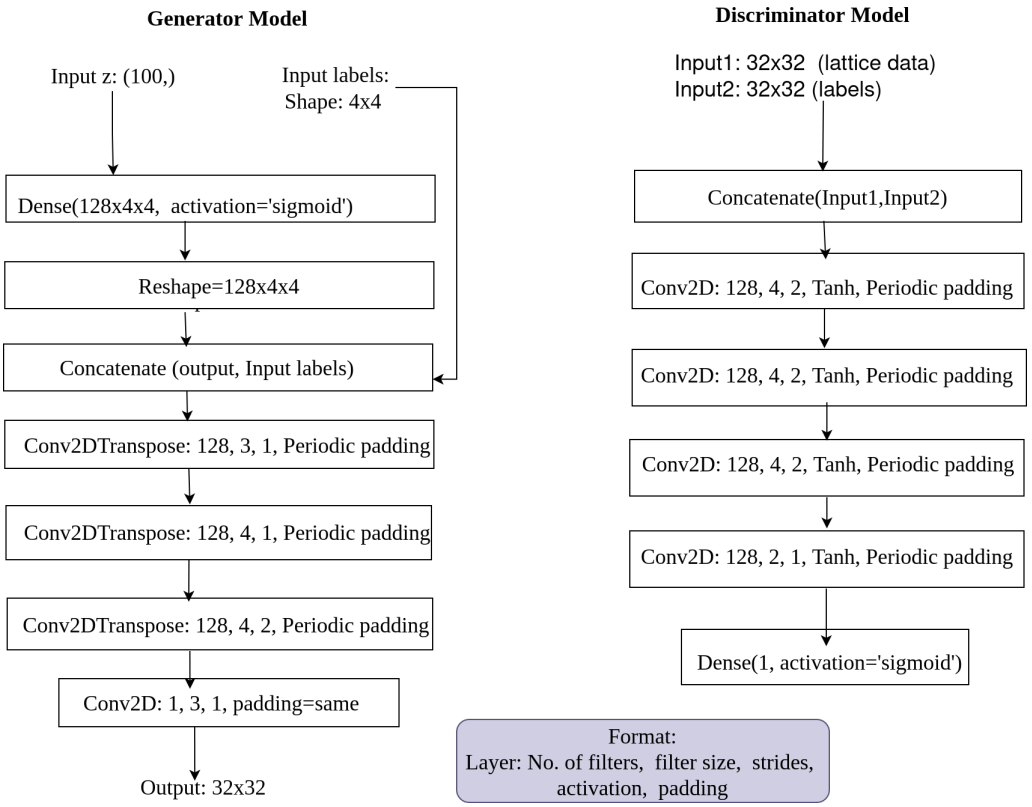


FIGURE 4.2: Architecture of C-GAN models: Generator and Discriminator model

to 32×32 final lattice configuration. We use the kernel of sizes (3,3) & (4,4) and strides (1,1) & (2,2) for the generator model. In the discriminator, the input is a grid of 32×32 σ samples, concatenated with a 32×32 channel with the λ value repeated in all the cells. It has three 2D convolutional layers with Tanh activation function, followed by a dense layer with Sigmoid activation. We use kernels of sizes (4,4) and strides (2,2), (1,1) for the discriminator. We add periodic padding to all layers of the generator model and only two initial layers of the discriminator model to learn the periodicity in the lattice configuration.

4.3.2 C-GAN training and sampling process

While training the generator, one batch of random noise z (with random λ) is drawn from $p_g(z)$. For training the discriminator, a batch of 256 configurations is used, where half the batch consists of $x = G(z|\lambda)$, where $z \sim p_g(z)$ and the other half consists of $x \sim p_{real}(x|\lambda)$. We use Adam optimizer with an initial learning rate of 0.0002 for both generator and discriminator loss. Initially, the C-GAN model is trained up to 200 epochs. As the stopping criteria for training, we use the observable error on the validation set comprising λ values from λ_{tr}. The observable error is computed as $\delta\sigma = \langle|\bar{\sigma}|\rangle_{hmc} - \langle|\bar{\sigma}|\rangle_{cgan}$. We stop

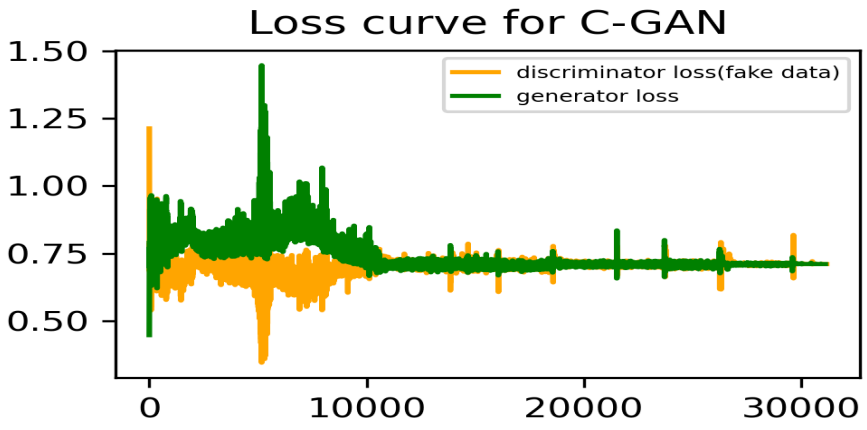


FIGURE 4.3: C-GAN loss curve: Interpolation case i.e. when both phases are used for training the C-NF model.

the training when this is minimum and doesn't reduce further. A plot of the training loss is shown Figure 4.3. The training becomes stable after $13k$ *or* $14k$ iterations.

For sampling, we choose a particular λ value of shape (4×4) and a batch from $z \sim p_g(z)$, fed to the pre-trained generator model. The batch size equals the number of samples required to generate, which is 2000 for our case. We generate 20000 configurations for different λ values.

4.4 Numerical Experiments and Results

We discuss two cases: interpolation and extrapolation to the critical region. In the interpolation, the C-GAN model is trained on a dataset from both sides of the critical region. In contrast, in extrapolation, we only use a training dataset from the symmetry-broken phase. We use the same model architecture and the same sampling procedures for both interpolating and extrapolating cases. The major difference is the size of the lattice ensemble used while training.

4.4.1 HMC simulation details

We employed HMC simulation for both the training dataset and the evaluation of C-GAN. In the HMC simulation, we simulate lattice configurations for $N_f = 2$, with lattice size=32×32. We adjust the MD step size to keep the acceptance rate around $\sim 80\%$ with legitimate autocorrelation time. We set the MD step size to 0.1 and the trajectory length

to 1. We discard the first 500 lattice configurations for thermalization. At each λ value in the range [0.6 - 2.5], we generate 4000 lattice configurations for the preparation of the training dataset.

4.4.2 Interpolation Case:

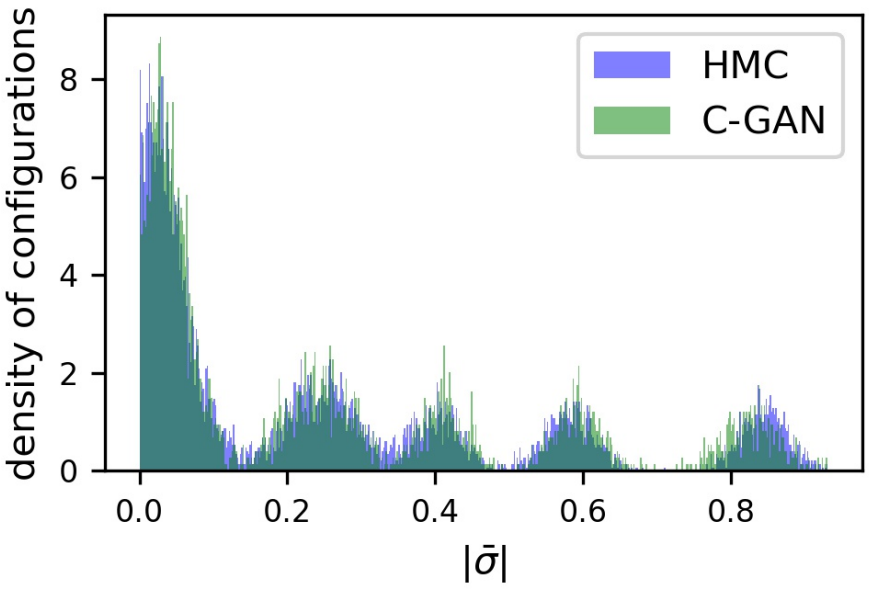
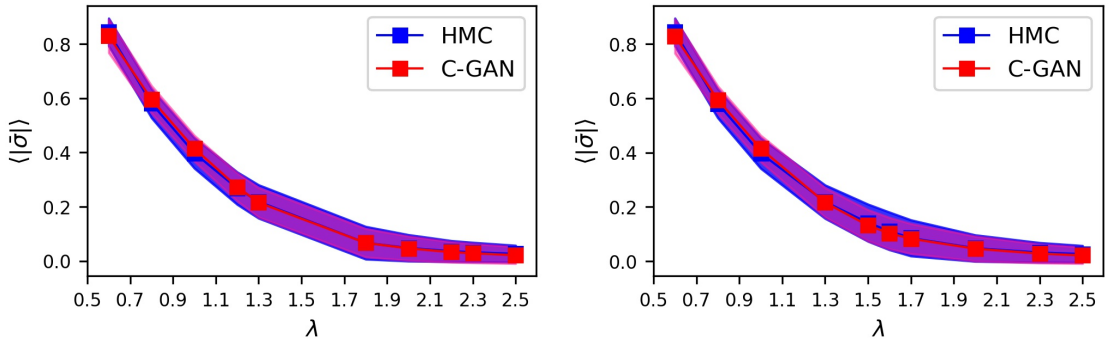


FIGURE 4.4: Histogram of $|\bar{\sigma}|$ for λ_{tr} set, estimated from samples obtained via HMC and C-GAN.



(A) (B)

FIGURE 4.5: a) Mean $\langle|\bar{\sigma}|\rangle$ and standard deviation for $\in \Lambda_{ts}$ and b) Λ_{ts} set,estimated from samples obtained via HMC and C-GAN

4.4.2.1 Dataset

In the case where we aim to perform interpolation, training is done on dataset consists of 10 ensembles, each of which has 4000 lattice configurations corresponding to 10 different λ values generated by HMC simulation. We denote Λ_{tr} as the set of λ on which we train the C-GAN model. It includes λ values away from the critical region. Assuming $\lambda_{crit} \sim 1.5$, $\Lambda_{tr} = \{0.6, 0.8, 1.0, 1.2, 1.3, 1.8, 2.0, 2.2, 2.3, 2.5\}$. For inference and evaluation of the proposed C-GAN model, we use Λ_{ts} which includes λ values in the critical region too, $\Lambda_{ts} = \{0.6, 0.8, 1.0, 1.3, 1.5, 1.6, 1.7, 2.0, 2.3, 2.5\}$.

We do the analysis for training λ values, i.e. λ_{tr} set to confirm that the C-GAN model has correctly learned the training data distribution.

In this ensemble, we calculate the $\bar{\sigma}$ for each lattice configuration, then plot the histogram of $|\bar{\sigma}|$ as shown in Figure 4.4. Different peaks in the histogram roughly correspond to different λ values. The histogram generated from the proposed C-GAN model and HMC overlaps quite well. It indicates that our proposed distribution $\hat{p}(\sigma|\lambda)$ represented by C-GAN approximates the true distribution for the λ_{tr} set. Also in Figure 4.5, we plot ensemble averaged $\langle|\bar{\sigma}|\rangle$ for Λ_{tr} set. Here we take ensemble average $\langle|\bar{\sigma}|\rangle$ for each λ separately and then plot $\langle|\bar{\sigma}|\rangle$ vs λ.

Since our main goal is to generate lattice ensembles in the critical region, we must evaluate our C-GAN model in the set Λ_{ts}. Again, we assess the performance of the proposed C-GAN in terms of being able to produce observables matching those obtained from true distributions(i.e., generated by HMC).

Mean$\langle|\bar{\sigma}|\rangle$: In Figure 4.6 we show the overlap of the histogram of $|\bar{\sigma}|$ from C-NF and HMC simulation. The peaks at high $|\bar{\sigma}|$ values roughly correspond to different λ values. However, there are no distinct peaks visible near low $|\bar{\sigma}|$ values as the peaks get overlapped. We also present the histograms of $|\bar{\sigma}|$ in Figure 4.7 for $\lambda \in \{1.5, 1.6\}$ which are in the critical region where we didn't train the model. In, Figure 4.5b we present the results for the mean $\langle\bar{\sigma}\rangle$ in the critical region. The results indicate that the trained model can reproduce the second-order phase transition in the GN lattice model.

Susceptibility(χ): We show the susceptibility values obtained from HMC configurations as well as those obtained from C-GAN in Figure 4.8a for the non-critical data set. One can observe that the peak coincides with both HMC and C-GAN. The same plot for the critical dataset is shown in Figure 4.8b.

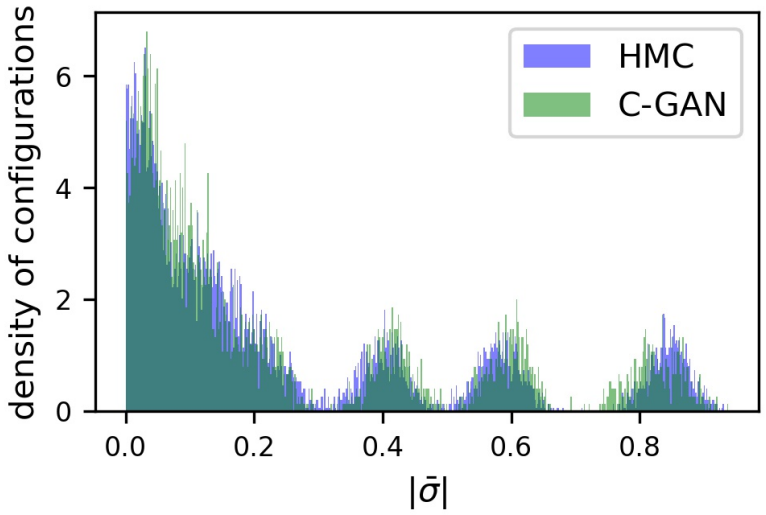
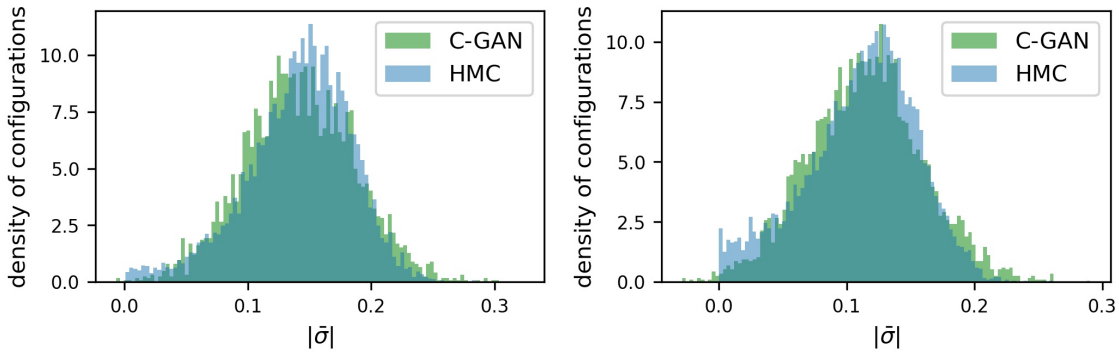


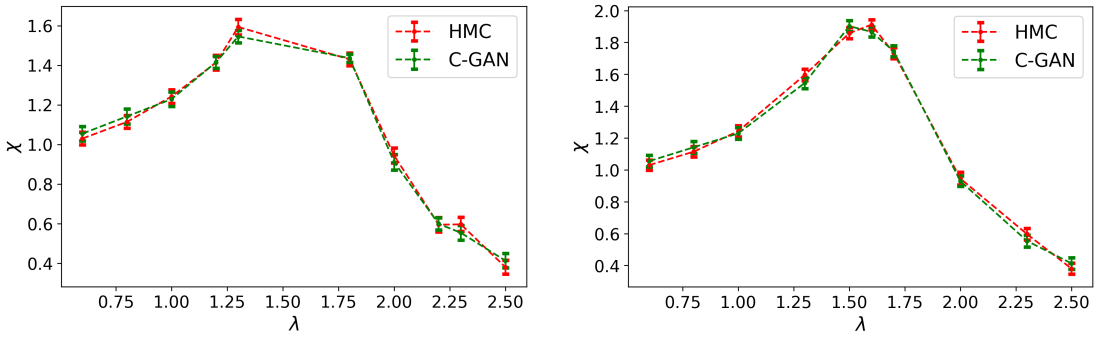
FIGURE 4.6: Histogram of $|\bar{\sigma}|$ for Λ_{ts} set, estimated from samples obtained via HMC and C-GAN.



(A) (B)

FIGURE 4.7: b) Histogram of $|\bar{\sigma}|$ at $\lambda = 1.5\&1.6 \in \Lambda^{ts}_{1ph}$: HMC and C-GAN histograms overlaps quite well.

In, Figure 4.9 we show the autocorrelation time generated from HMC simulation with unit trajectory length in the MD step while keeping acceptance rate $\approx 80\%$. We see that near the phase transition point, the autocorrelation time increases sharply. However, during sampling from the C-GAN model, we start with a random Gaussian noise vector to generate lattice configurations. Therefore, the lattice configurations generated by the C-GAN model are independent of each other, which will solve the critical slowing down problem. In this way, we can generate uncorrelated samples near the critical region at the cost of the generation of samples by HMC at the non-critical region.



(A) (B)

FIGURE 4.8: Susceptibility and its standard deviation estimated from 8000 samples with bin size of 100 for a) λ_{tr} set and b) $\in \Lambda_{ts}$.

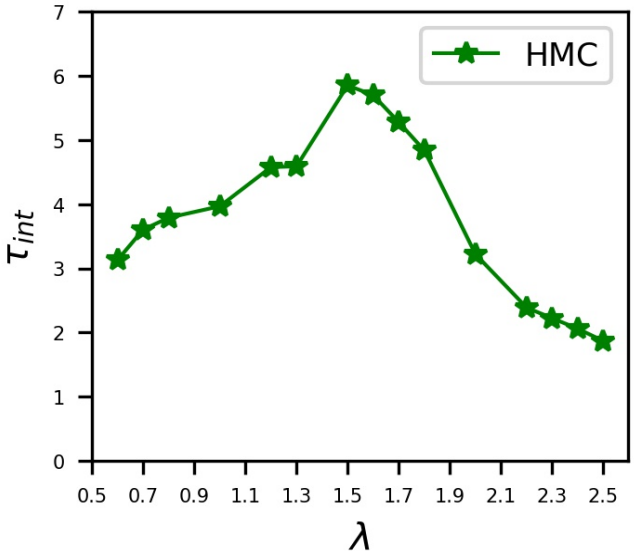


FIGURE 4.9: Integrated autocorrelation time from HMC simulation for $\langle|\bar{\sigma}|\rangle$

.

4.4.3 Extrapolation Case:

We also train the C-GAN model using an HMC-generated dataset consisting of λ values only from one single phase. This experiment is necessary to check our model's utility in lattice gauge theory, where extrapolation to critical points is necessary from one direction of parameter space.

For extrapolation purpose, we choose the training set of λ values as λ^{tr}_{1ph} ={0.4, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.25, 1.3, 1.35, 1.4} taken from the broken phase and the test set of λ

values are λ^{ts}_{1ph}= {0.7,0.9, 1.0, 1.2, 1.3, 1.45, 1.5, 1.55, 1.6}. For each $\lambda \in \lambda^{ts}_{1ph}$ we generate 60000 samples for training the C-GAN from the HMC simulation. However, the model architecture and training hyperparameters such as batch size, learning rate etc. remains same as in the interpolation case. We extrapolate the C-GAN model to the critical region of λ values 1.45, 1.5, 1.55 and 1.6.

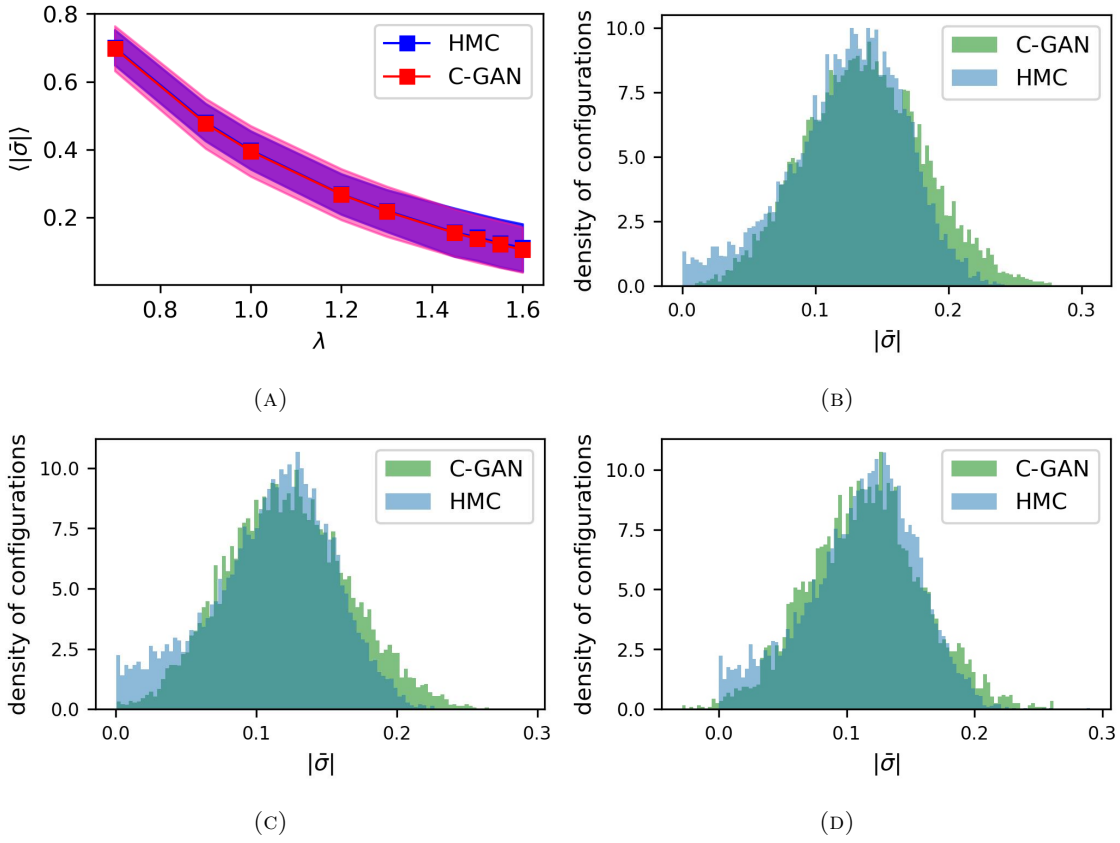


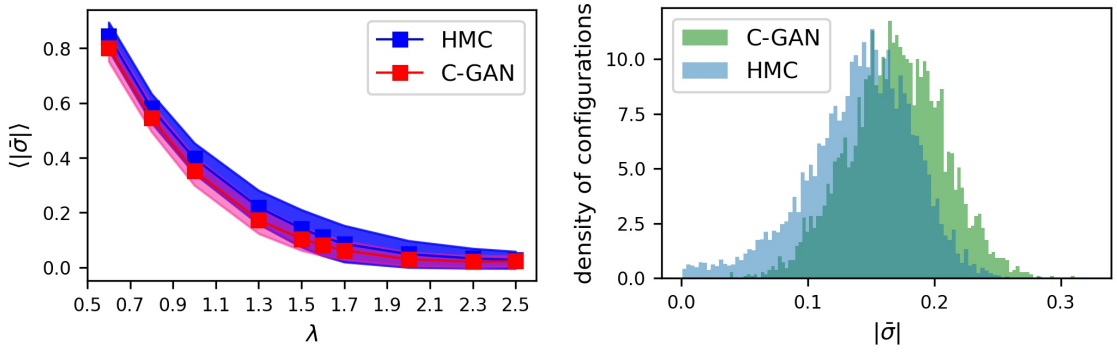
FIGURE 4.10: a) Mean $\langle|\bar{\sigma}|\rangle$ and standard deviation for Λ^{ts}_{1ph} set, estimated from samples obtained via HMC and C-GAN. b) Histogram of $|\bar{\sigma}|$ at $\lambda = 1.6 \in \Lambda^{ts}_{1ph}$: HMC and C-GAN histograms overlaps quite well.

We compare few observables estimated using C-GAN and HMC simulations. The results are shown in Figure 4.10 for λ^{ts}_{1ph}, where critical points are included. We observe that the histogram and mean $\langle\bar{\sigma}\rangle$ match well within statical uncertainties. We found that for the critical λ values observables don't differ, either we train the model with one single phase or consider both phases.

In Figures 4.7a, 4.10b and 4.10c we have shown the individual histogram of $|\bar{\sigma}|$ for λ=1.5, 1.55, 1.6. respectively.

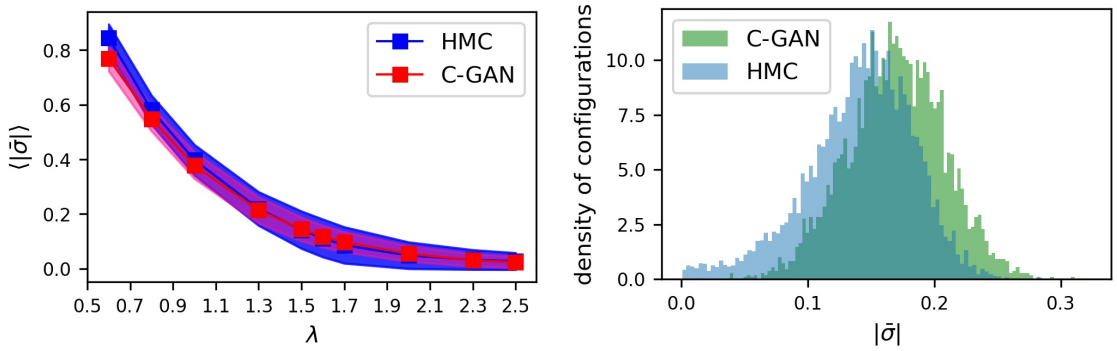
4.4.4 Ablation Analysis

We perform an ablation analysis to see the effect of certain key components of the proposed method on its performance.



(A) (B)

FIGURE 4.11: a) Ablation for log transformation, we plot mean $\langle|\bar{\sigma}|\rangle$ and standard deviation on Λ_{ts} set without log transformation. b) Ablation for log transformation: Histogram of $|\bar{\sigma}|$ at λ =1.5 $\in \Lambda_{ts}$ set without log transformation.



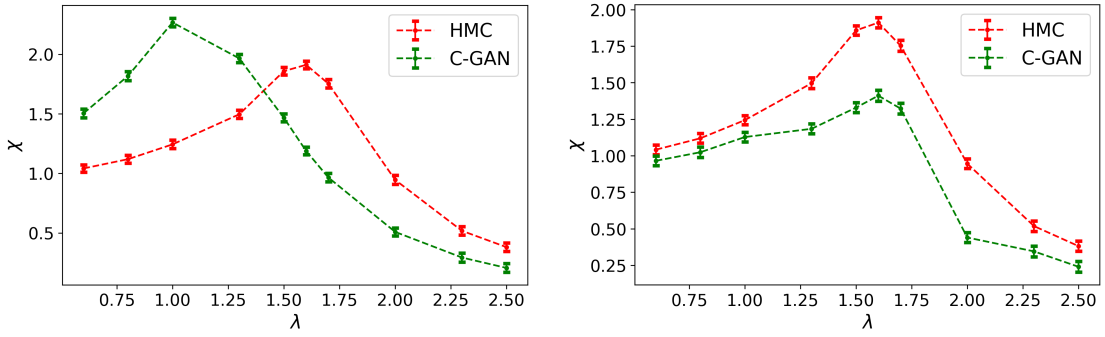
(A) (B)

FIGURE 4.12: a)Ablation for periodic padding: without periodic padding in both generator and discriminator. We plot to Mean $\langle|\bar{\sigma}|\rangle$ and standard deviation on Λ_{ts} set without log transformation. b) without periodic padding in both the generator and discriminator.

Transformation of σ field: We find that log transformation Equation (4.16) is one of the crucial components for the training of the C-GAN model. When removed, the training loss becomes high and the observables do not agree well with the HMC observables which can be seen from Figure 4.11. There is a large deviation of the mean of $|\bar{\sigma}|$ compared to HMC results in both critical and non-critical regions, as shown in Figure 4.11a. It is observed that susceptibility is too sensitive to log transformation, as shown in Figure 4.13a.

It is also seen that without Min-Max scaling, the C-GAN model is unable to learn the different modes corresponding to different λ values.

Periodic Boundary Condition: We have noticed that the C-GAN performs well using periodic padding in both discriminator and generator, as seen from Figure 4.5. But when we remove periodicity from both the discriminator and generator, C-GAN fails to reproduce the HMC results. Figure 4.12 show the disagreements between C-GAN and HMC results for $\langle|\bar{\sigma}|\rangle$, $|\bar{\sigma}|$ respectively and Figure 4.13b compares the susceptibility χ without periodicity in C-GAN for $\lambda = 1.5$. Likewise, not applying periodic padding to the generator and applying only to the discriminator, also degrades the performance.



(A) (B)

FIGURE 4.13: a)Ablation for log transformation: Susceptibility on Λ_{ts} set without log transformation b) Ablation for periodic padding: Histogram of $|\bar{\sigma}|$ at λ =1.5 $\in \Lambda_{ts}$ set without periodic padding in both generator and discriminator.

4.4.5 Cost Analysis

Sampling a fixed number of configurations via the HMC algorithm depends on various parameters like MD step size, no. of MD steps, the parameter value of action and also hardware used for simulations. Here we have used MD stepsize=0.1 and no. of MD steps=10. The hardware used for HMC simulations is a six-core i7-9700CPU CPU machine. In the non-critical region, we generate around 10000 configurations per hour, leaving 10 intermediate configurations. In the critical region, we leave 20 intermediate configurations and are able to generate roughly 4000 configurations per hour. The training of the C-GAN model was done on a single GPU machine (GeForce RTX) for 2-3 hours[1]. Once the training is over, sampling of lattice configurations becomes very efficient. It roughly takes only 2 minutes to generate 8000 lattice configurations.

[1]We have used TensorFlow 2.4 for our model implementation.

It should be noted that comparing the costs of HMC and C-GAN is not a straightforward task due to their distinct sampling algorithms. Additionally, the presence of autocorrelation in the Gross-Neveu model does not pose a major problem on a 2D lattice. The objective of our study is to evaluate the ML-based approach in a less complex scenario. Nevertheless, the observed gain appear substantial, and we anticipate even greater advantages in higher dimensions, where autocorrelation in HMC simulations becomes more pronounced near the critical region.

4.5 Summary

We discussed a GAN-based generative model to learn a conditional distribution over the parameter value of the Gross-Neveu model in 2D. We have trained the model with samples from the non-critical region and extrapolated/interpolated the model in the critical region. In the critical region, we performed our analysis on the configurations generated by the C-GAN and showed different results and compared them with baseline HMC simulation. We test our approach for the Gross-Neveu model in the 1+1 dimension. We find that the observable distributions obtained from the proposed C-GAN model match with those obtained from HMC simulations while circumventing the problem of critical slowing down.

Chapter 5

Flow-based Conditional Model for Sampling a Lattice Distribution

In the previous chapter, a conditional GAN (C-GAN) [115] has been interpolated in the critical region of the Gross-Neveu model for generating lattice samples. Although empirically, we have found a good agreement between various observable estimates from both HMC and C-GAN, there was no underlying theoretical guarantee that the generated samples are coming from the target lattice distribution. This is mainly because generative models learn the target distribution approximately by minimizing some metric between the target and model distribution. Therefore, there could be biased in the observable if the model is not trained perfectly. On the other hand, if a generative model can estimate the exact density of the generated samples, then the model could be used as a proposal density to construct a Markov chain which will provide the theoretical guarantee of exactness. GAN does not provide the probability density of generated samples; hence, we can not provide the theoretical guarantee of exact sampling in the C-GAN model.

Normalizing Flow (NF) [116] are generative models which can generate samples from the target distribution and estimate the generated sample's density. Therefore, an NF model can be employed as a proposal distribution to construct a Markov chain. We develop a provably correct Conditional Normalizing Flow-based (C-NF) [117] sampling method for Scalar ϕ^4 theory. In this chapter, we will discuss the construction of a conditional NF model (C-NF), which can learn a conditional distribution across the action parameter of the ϕ^4 theory in the $1+1$ dimension. Once trained on the non-critical regions, the C-NF model can be interpolated or extrapolated in the critical region. In the critical region, we will use the independent Metropolis Hasting (MH) algorithm [60] to remove the artefact

in the model, where the C-NF model will act as a proposal density. This approach allows us to generate different lattice ensembles for multiple action parameters from the same model.

In ϕ^4 theory, the goal is to generate lattice configurations according to the distribution

$$p_\Phi(\phi|\lambda) = \frac{1}{Z} e^{-S(\phi|\lambda)}, \tag{5.1}$$
$$\text{where, } Z = \int D\phi e^{-S(\phi|\lambda)}.$$

Here, ϕ is the lattice field and Z is the partition function. The S represents the lattice action of ϕ^4 theory defined as

$$\begin{aligned} S(\phi, \lambda, m) = &\sum_x \Big[\sum_{\mu=1,2} \phi(x) \Big[2\phi(x) - \phi(x+\hat{\mu}) \\ &- \phi(x-\hat{\mu}) \Big] + m^2 \phi(x)^2 + \lambda \phi(x)^4 \Big], \end{aligned} \tag{5.2}$$

where, x is a $2d$ discrete vector, and $\hat{\mu}$ represents two possible directions on the lattice. The lattice theory has two dimensionless parameters λ and m. In our construction, we fixed one parameter, (we fix $m^2 = -4$) and hence we omit this parameter in the expression of conditional distribution $p_\Phi(\phi|\lambda)$.

We train a C-NF model using HMC samples from $p_\Phi(\phi|\lambda)$ for $\lambda \in \Lambda_{NC}$, where Λ_{NC} denotes the non-critical region[1] of the theory. We are particularly interested in lattice configurations for $\lambda \in \Lambda_C$, where Λ_C denotes the critical region of the theory.

As earlier, here we study two kinds of cases, one where non-critical regions exist on both sides of the critical point and the other where non-critical region exists only on one side. We apply C-NF to generate lattices for both cases. We interpolate the C-NF model for the former case, and for the latter, we extrapolate the model to the critical region. Interpolation is useful for studying phase transition in a statistical system, while extrapolation is useful for obtaining the continuum limit in LFT.

5.1 Conditional Normalizing Flow

In Chapter 3, we have discussed Normalizing flow models and the construction of flows like affine coupling flow. Now, we will discuss how to introduce a conditional parameter into an

[1] we define critical and non-critical sets based on integrated autocorrelation time.

NF model to learn a conditional distribution. Conditioning the parameter is not straight forward as conditioning a GAN due to the requirement of invetibility of an NF model. There are different ways of conditioning an NF model. To gain a better understanding, one can see the discussion of affine transformation as we discussed in Section 3.2.2. It is clear from the Figure 5.1 that the conditional parameter can be given as input to the neural network **s** and **t**. This will not alter the invertibility of the model as **s** and **t** are computed only in the forward direction.

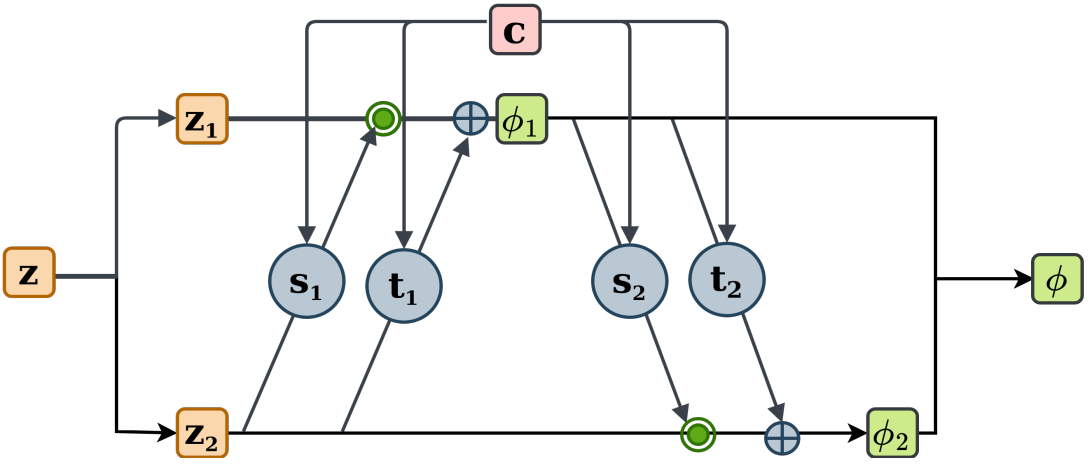


FIGURE 5.1: One affine block of Conditional Normalizing flow, where s and t are convolutional neural networks.

The transformation rules for such a affine coupling block are

$$\begin{aligned} \phi_1 &= z_1 \odot e^{(s_1(z_2,c))} + t_1(z_2, c), \\ \phi_2 &= z_2 \odot e^{(s_2(\phi_1,c))} + t_2(\phi_1, c), \end{aligned} \tag{5.3}$$

Where $\odot$ represents the element-wise product of two vectors, for conditional learning, we concatenate the conditional parameter c along with the input z_2 and feed into the convolutional networks **s** and **t** [118] as shown in Figure 5.1. We combine many such coupling blocks to construct the C-NF model. Note that, we have not conditioned each coupling layer of the C-NF model with λ. In numerical experiments, we found that if we condition each model layer with λ, the model overfits the λ values used for training and does not generalize to interpolated/extrapolated λ values. For the results shown, we condition only the layers with index $n \times k$, where $n = 0, 1, ...$ and $k = 1, 5, 9$ for lattices of size $(6 \times 6$ & $8 \times 8)$, 12×12 and 16×16, respectively. The model descriptions are shown in the Figure 5.2.

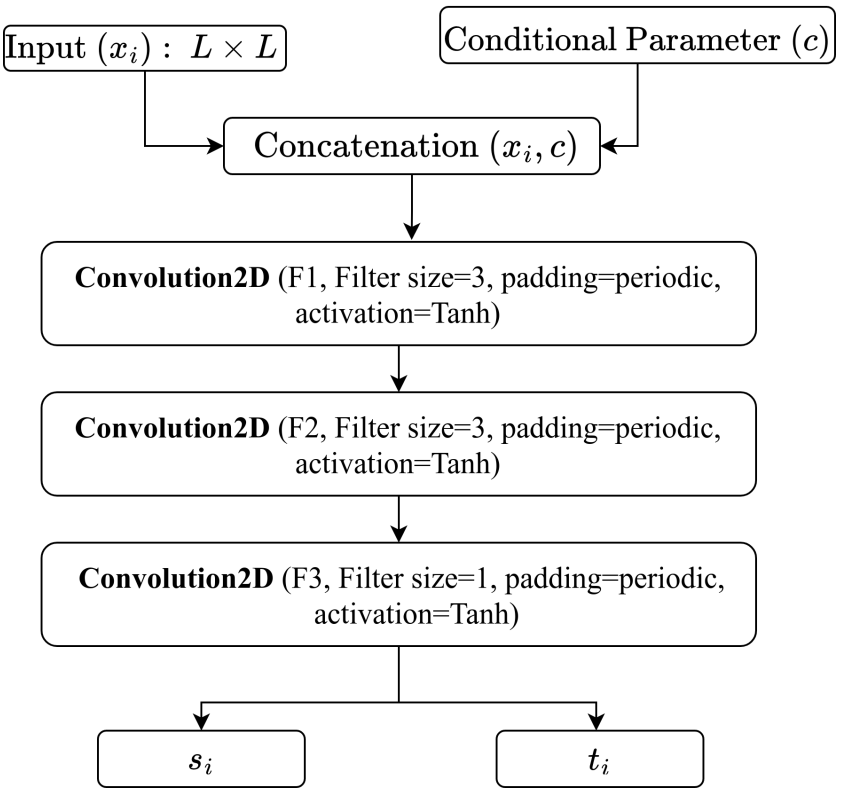


FIGURE 5.2: Architecture of the Neural Networks **s** and **t** of the i-th affine coupling layer.

Let us designate the C-NF model as $f(\phi; c, \theta)$ and it's inverse as $g(z; c, \theta)$. The invertibility for any fixed condition c is given by

$$f^{-1}(.; c, \theta) = g(.; c, \theta). \tag{5.4}$$

Sample generation After training, for a fixed c, we can execute conditional generation of ϕ by sampling z from a simple prior distribution i.e. $z \sim p_Z(z)$. Now, use the inverted network (trained) $g(z; c, \theta_{ML})$ to generate required samples i.e. x=$g(z; c, \theta_{ML})$.

Density estimation To estimate the probability density function of generated samples, we can directly use

$$q_\Phi(\phi; c, \theta) = p_Z(f(\phi; c, \theta)) \Big| \det \frac{\partial f_\theta^{-1}(\phi)}{\partial \phi} \Big|. \tag{5.5}$$

5.2 Application to Lattice Scalar ϕ^4 Theory

The lattice action as in Equation (5.2) possesses $\phi(x) \to -\phi(x)$ symmetry. However, it is spontaneously broken at a specific parameter region. For the numerical investigation, we consider only the phase transition w.r.t. parameter λ. For a given λ, the probability distribution of lattice configuration is given by $p_\Phi(\phi|\lambda)$ defined in Equation (5.1).

More information on lattice ϕ^4 theory can be found in [119]. Some of the observables which we will be calculating on the lattices are:

1. $\langle\tilde{\phi}^2\rangle$: $\tilde{\phi} = \frac{1}{V}\sum_x \phi(x)$.
2. Zero momentum Correlation Function:

$$C(t) = \sum_{x_1} G_c(x_1, t),$$

   where, $x = (x_1, t)$ and $G_c(x) = \frac{1}{V}\sum_y[\langle\phi(y)\phi(x+y)\rangle - \langle\phi(y)\rangle\langle\phi(x+y)\rangle]$.
3. Two Point Susceptibility: $\chi = \sum_x G(x)$.

In the C-NF model, ϕ in Equation (5.3) corresponds to the lattice field configuration of scalar ϕ^4 theory. The action parameter λ represents the conditional parameter c. λ is fed to the network as a separate channel appended to z with values $\lambda\mathbb{1}$, where $\mathbb{1}$ is a matrix with all entries as identity and size same as that of z. We generate training samples ϕ from the distribution defined in Equation (5.1):

$$z = g(\phi; \lambda, \theta) \quad ; \qquad \phi \sim p_\Phi(\phi|\lambda). \tag{5.6}$$

The loss function for optimizing the C-NF model is the forward KL divergence between the model distribution $q_\Phi(\phi; \lambda, \theta)$ and target distribution $p_\Phi(\phi; \lambda)$:

$$\begin{aligned}\mathcal{L}(\theta) &= \int d\phi\ p_\Phi(\phi; \lambda)(\log(p_\Phi(\phi; \lambda)) - \log(q_\Phi(\phi; \lambda, \theta))), \\ &= E_{\phi\sim p_\Phi}[\log(p_\Phi(\phi; \lambda)) - \log(q_\Phi(\phi; \lambda, \theta))].\end{aligned} \tag{5.7}$$

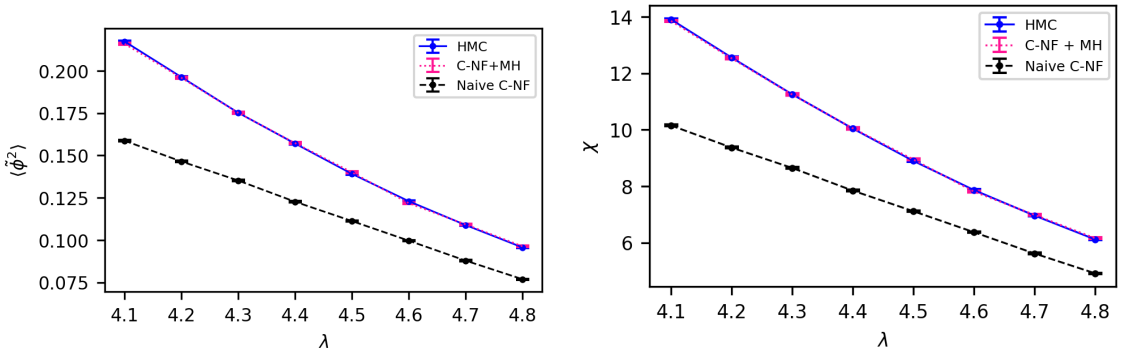
5.3 Training and Sampling Procedure

5.3.1 Training

We train two models for studying the two different cases: interpolation and extrapolation to the critical region. We train these models by using HMC-generated samples for $\lambda \in \Lambda_{NC}$. In HMC simulation, we use Molecular Dynamics (MD) step size=0.1 and MD trajectory length=1. For each $\lambda \in \Lambda_{NC}$, we generate 10,000 lattice configurations for the interpolation case and 15,000 for the extrapolation case. We use a fixed number of training configurations rotated throughout the training process for both cases.

For the estimation of observables from HMC, we generate 10^5 lattice configurations. We use the bootstrap re-sampling method to estimate the observables' uncertainty with a bin size of 100.

To define the sets Λ_C and Λ_{NC}, we choose two threshold values of λ on both phases based on integrated autocorrelation time (τ) of χ. We select the threshold value of λ such that the maximum τ for Λ_{NC} set is 1.5.



(A) $\langle\tilde{\phi}^2\rangle$ (B) χ

FIGURE 5.3: Interpolation to the critical region:-$\langle\tilde{\phi}^2\rangle$ and χ are calculated on samples generated from i)HMC, ii)C-NF followed by MH, and iii) Naive C-NF. The Error bars indicate the standard deviation calculated using bootstrap re-sampling with bin size 100.

During training, we use a batch size of 1024, with each sample consisting of (ϕ, λ), chosen randomly from the ensembles of different λ values. We employ an Adam optimizer with an initial learning rate of 0.0003.

5.3.2 Sampling

For sampling, we draw z from a prior distribution which we set as standard normal, $z \sim \mathcal{N}(0, 1)$. We feed z into the C-NF model to generate lattice samples $\phi = f(z)$, which are used as proposal samples for constructing a Markov chain. We choose a $40 - 45\%$ acceptance rate as the stopping criteria for training. For the C-NF model, we get the target acceptance rate of around 65k training iterations. Once training is over, we interpolate or extrapolate the model to the critical region. We give critical λ values as conditional parameters for generation in the critical region. We use the acceptance rate, Effective Sample Size (ESS) and three observables $\langle\tilde{\phi}^2\rangle$, χ and $C(t)$ as metric to evaluate the performance of the C-NF model. We calculate the observables for four different lattice sizes 6×6, 8×8, 10×10 and 12×12.

5.3.3 C-NF Model Quality: Effective Sample Size (ESS)

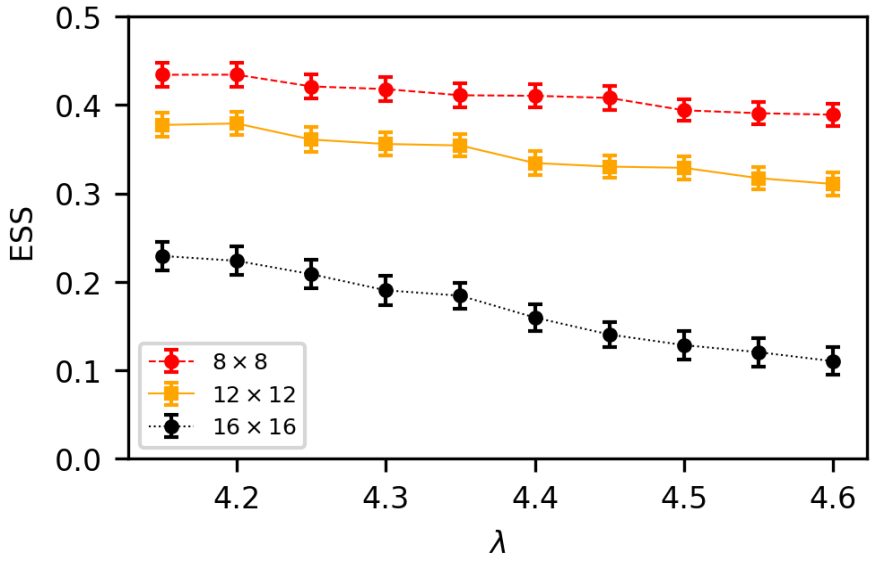


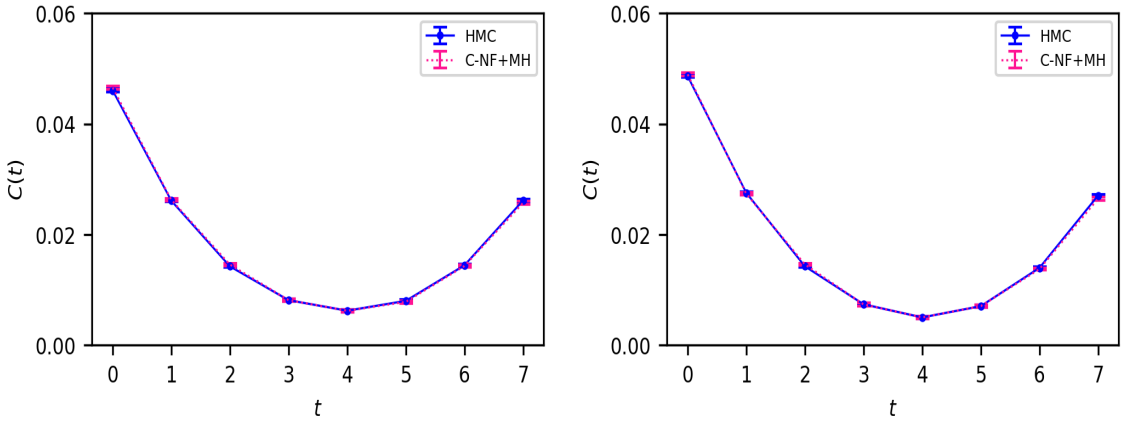
FIGURE 5.4: ESS for extrapolated λ calculated from models trained on three different lattice volumes. For each lattice, we use 65k iterations for training.

The ESS is defined as [63]

$$ESS = \frac{1}{N} \frac{\left(\sum_{i=1}^{N} p_\Phi(\phi_i; \lambda)/q_\Phi(\phi_i; \lambda, \theta)\right)^2}{\sum_{i=1}^{N} \left(p_\Phi(\phi_i; \lambda)/q_\Phi(\phi_i; \lambda, \theta)\right)^2}. \tag{5.8}$$

We estimate ESS for three different C-NF models corresponding to three lattice sizes, namely, 8×8, 12×12, 16×16. For the estimation of ESS, we use a batch size of 5000

lattice configurations from each model trained to 65k iterations. We took 100 independent estimations of ESS from each model to calculate its uncertainty. In Figure 5.4, we plot the ESS for 10 different λ values for each model. The ESS is almost constant for the extrapolated λ values for 8 and 12×12 lattices. For 16×16, the ESS gradually decreases to 10% at λ values close to the critical point.



(A) $\lambda = 4.5$ (B) $\lambda = 4.70$

FIGURE 5.5: Interpolation: Zero momentum correlation function $C(t)$ calculated on samples generated from i)HMC and ii)C-NF followed by MH. The error bars indicate the standard deviation calculated using bootstrap re-sampling with bin size 100.

5.4 Numerical Experiments and Results

5.4.1 Interpolation to the Critical Region

For interpolation case, we choose training Λ_{NC} set as: $\{3, 3.2, 3.5, 3.6, 3.7, 3.8, 5.8, 6, 6.5, 7, 8, 9\}$, with threshold λ=3.8 and 5.8. We interpolate the trained model for multiple λ values, $\Lambda_C : \{4.1, 4.2, 4.25, 4.3, 4.35, 4.4, 4.45, 4.5, 4.55, 4.6, 4.65, 4.7, 4.8, 5.0\}$. For each λ value, we generate a Markov chain of 10^5 configurations from the proposed method. Then observables are calculated on each ensemble for both phases around the critical point. We compare the observables from our proposed method with those from HMC simulation in Figures 5.3, 5.5, 5.8, 5.9 and 6.1. Observables from the naive C-NF without the MH algorithm are also shown to illustrate the C-NF model's proximity to the true distribution. In Figure 5.3, we plot the observables $\langle\tilde{\phi}^2\rangle$ and χ for $\lambda \in \Lambda_C$. In Figure 5.5, the two-point zero momentum correlation function $C(t)$ is shown for two λ values. We observe a larger correlation for these λ values compared to non-critical λ values.

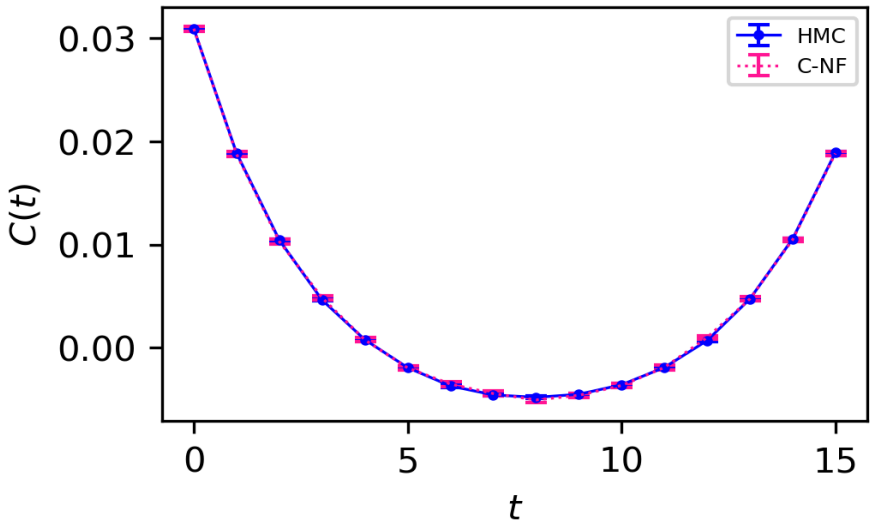
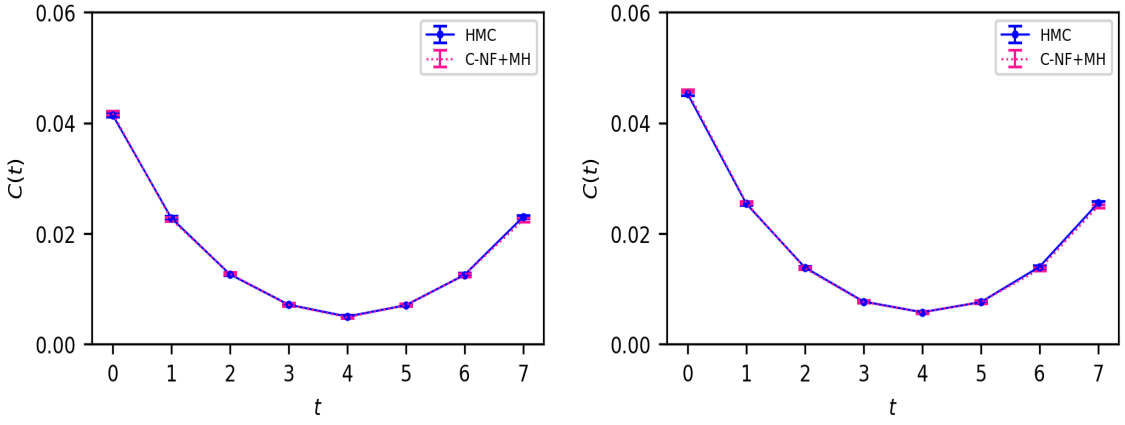


FIGURE 5.6: Zero momentum correlation function $C(t)$ calculated on 16×16 lattice samples generated from i)HMC and ii)C-NF followed by MH. The error bars indicate the standard deviation calculated using bootstrap re-sampling with bin size 100.

In Figure 5.9, we also show the histogram of $\tilde{\phi}$ for a particular critical $\lambda = 4.6$. It visually demonstrates the elimination of artefacts by the MH algorithm, which the C-NF model produced.

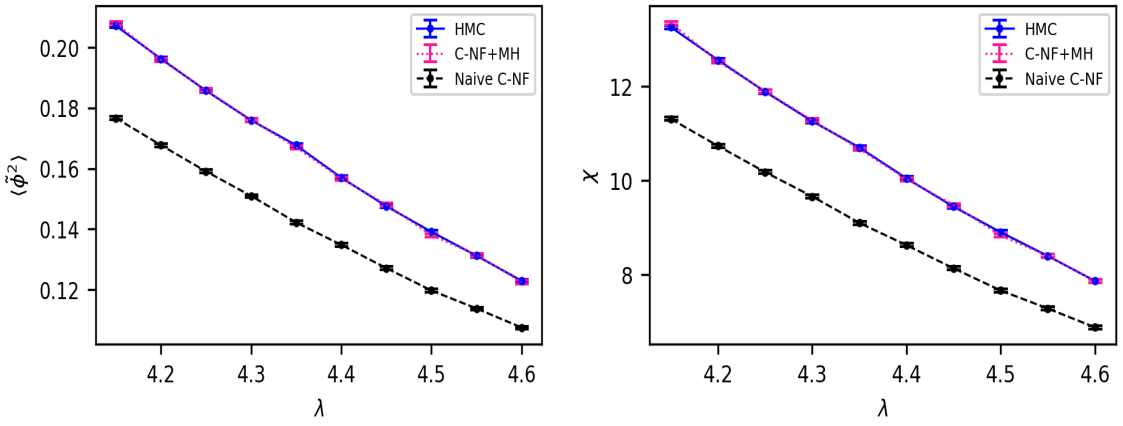


(A) $\lambda = 4.30$ (B) $\lambda = 4.45$

FIGURE 5.7: Extrapolation:- Zero momentum Correlation function calculated on samples generated from i)HMC and ii)C-NF followed by MH. The error bars indicate the standard deviation calculated using bootstrap re-sampling with bin size 100.

5.4.2 Extrapolation to the Critical Region

The Λ_{NC} set used to train the C-NF model for the extrapolation case in the broken phase is $\{3, 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9\}$. We find that in this case, the size of the training dataset has to be increased to achieve ESS comparable to the interpolation case. After training, we extrapolate it in the critical region around $\lambda = 4.6$, which is taken as the critical region's midpoint i.e. $\lambda \in \{4.2, 4.3, 4.4, 4.5, 4.6\}$. For each λ value, we generate one ensemble of 10^5 configurations from our method.



(A) $\langle\tilde{\phi}^2\rangle$ (B) χ

FIGURE 5.8: Extrapolation to the critical region: $\langle\tilde{\phi}^2\rangle$ and χ are calculated on samples generated from i)HMC, ii)C-NF followed by MH, and iii) Naive C-NF. The error bars indicate the standard deviation calculated using bootstrap re-sampling with bin size 100.

We plot the observables $\langle\tilde{\phi}^2\rangle$ and χ in Figure 5.8 and the zero momentum correlation function is plotted in Figure 6.1 for two different $\lambda \in \Lambda_C$.

Integrated Autocorrelation time for $\langle\tilde{\phi}^2\rangle$ from the extrapolated C-NF model without applying MH is shown in Table 5.1. Also, the two observables χ and $\langle\tilde{\phi}^2\rangle$ from interpolated C-NF are shown in Table 5.2.

λ	4.2	4.3	4.4	4.5	4.6
τ_{int}	1.046	1.049	1.027	1.032	1.010

TABLE 5.1: Integrated Autocorrelation time for $\langle\tilde{\phi}^2\rangle$ from the extrapolated C-NF model without applying MH.

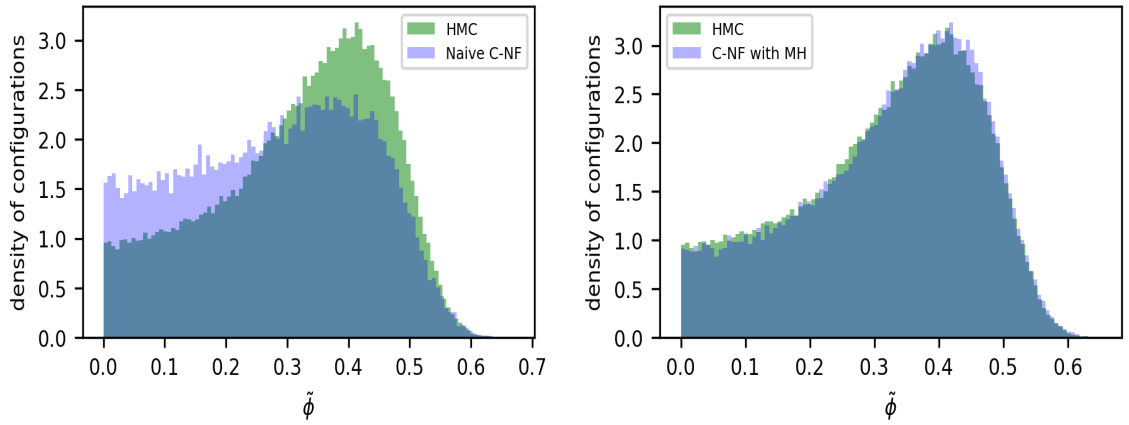


FIGURE 5.9: Histogram of $\tilde{\phi}$ for $\lambda = 4.6$ from the a)naive C-NF model and b) C-NF with MH is compared against HMC results with a bin-size=100.

λ	$\langle\tilde{\phi}^2\rangle$			χ_2		
	HMC	C-NF with MH	C-NF without MH	HMC	C-NF with MH	C-NF Without MH
4.10	0.2173 ± 0.000560	0.2158 ± 0.000613	0.1586 ± 0.004405	13.9088 ± 0.035509	13.8540 ± 0.033144	10.1537 ± 0.028371
4.20	0.1962 ± 0.000535	0.1958 ± 0.000682	0.1463 ± 0.004499	12.5538 ± 0.037975	12.5331 ± 0.045937	9.3705 ± 0.028212
4.30	0.1751 ± 0.000595	0.1751 ± 0.0006534	0.1361 ± 0.004499	11.2590 ± 0.036719	11.2494 ± 0.040154	8.6496 ± 0.028159
4.40	0.1570 ± 0.000588	0.1572 ± 0.000689	0.1226 ± 0.00387	10.0473 ± 0.036727	10.0576 ± 0.040644	7.8483 ± 0.024385
4.50	0.13911 ± 0.000564	0.1400 ± 0.000601	0.11226 ± 0.003910	8.9027 ± 0.034886	8.9507 ± 0.0355522	7.1191 ± 0.025168
4.60	0.1229 ± 0.000556	0.1217 ± 0.000523	0.0996 ± 0.003718	7.8695 ± 0.031900	7.8145 ± 0.035491	6.3767 ± 0.023685
4.70	0.1088 ± 0.000504	0.1092 ± 0.000487	0.0878 ± 0.003412	6.9668 ± 0.03192	6.9917 ± 0.03131	5.6231 ± 0.0218635
4.80	0.0956 ± 0.000452	0.0963 ± 0.000463	0.0768 ± 0.003276	6.1230 ± 0.028696	6.164447± 0.030102	4.9206 ± 0.021078

TABLE 5.2: Two observables calculated on samples generated from i) HMC ii) C-NF with MH and iii) Naive C-NF in the interpolation case. The error indicates the standard deviation calculated using bootstrapping re-sampling with bin size 100.

The naive C-NF model produces inherently uncorrelated lattice configurations. However, using such configurations directly from the naive C-NF model will lead to biases in the observables. We use the independent MH algorithm to avoid the model's biases, which guarantees the asymptotically exact Markov chain.

5.4.3 Comparison to a Non-Conditional NF Model

A non-conditional NF model trained on a single λ value can be also used to generate proposals for the MH algorithm to construct ensembles for other λ values. However, this model may not generate good proposals for the target distribution. Here we compare the efficiency of such a model with C-NF models. The efficiency will depend on the acceptance rate in the MH algorithm for any non-training λ. We train a non-conditional NF with

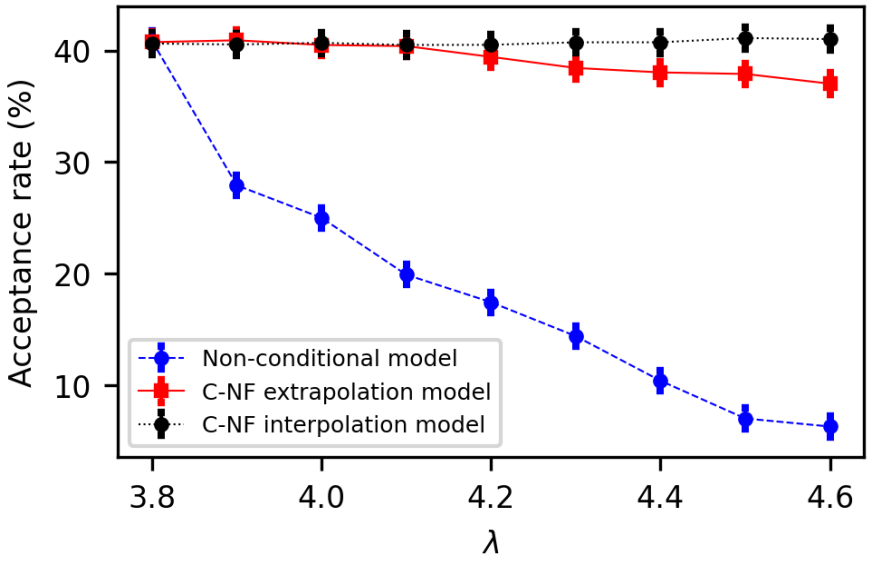


FIGURE 5.10: MH acceptance rate from three different models are shown: (1) model trained on a single $\lambda = 3.8$ (blue dashed); (2) C-NF extrapolated model (red solid) and (3) C-NF interpolated model (black dotted).

samples for a single $\lambda = \lambda_0 = 3.8$ and use it to generate proposals for the MH algorithm at other λ values. In Figure 5.10 we compare the acceptance rate for three different models. We observe a faster decline in the acceptance rate as we move away from λ_0 as compared to the acceptance rate obtained with the interpolated and extrapolated C-NF.

5.5 Cost Analysis

The sampling algorithm for the baseline approach (HMC) and the suggested method is vastly different; thus, a direct cost comparison is opaque. Nonetheless, we separate the simulation cost for the proposed technique into two components: training time and sample generation time. On a Colab, the training time for the C-NF model is roughly 5-6 hours. However, sample generation is very fast for the C-NF model once training is over. Generating a Markov chain of 10^5 configuration from C-NF+MH takes only 5-7 minutes with $\sim 40\%$ acceptance rate. Due to the short generation time, it can be utilized for the generation of large ensembles of lattice configurations. The proposed method allows us to extend the same model to generate lattice samples for any value of action parameters λ in the range $[3.0 - 10]$.

Therefore, the proposed method could be efficient when lattice simulation at multiple λ values, especially in the critical region, is required.

5.6 Summary

In this chapter, we investigated the effectiveness of Normalizing flows in sampling lattice distributions through the construction of a Markov chain. Our research demonstrated the promising potential of generative machine learning techniques, particularly normalizing flows, in speeding up MCMC simulations, specifically in critical regions of LFT. We addressed the challenge of training these models for different parameter values within the critical region by proposing a conditional model trained in the non-critical region, with subsequent interpolation or extrapolation to extend its applicability to any parameter value within the critical region. We ensured the asymptotic exactness of the Markov Chain through the employment of the MH algorithm. We tested our proposed method for MCMC sampling in critical regions of $(1+1)$-dimensional ϕ^4 theory. As we move forward, one significant challenge we aim to explore is the scalability of the C-NF model, enabling accurate estimation of physical quantities in near-critical regions.

Chapter 6

An Equivariant Conditional Flow Model for Sampling Gauge Theory

In this chapter, we will discuss the application of the conditional flow-based model for sampling U(1) gauge theory in two dimensions.

In lattice Gauge theory, such as $U(1)$ gauge theory in (1+1) dimensions, Monte Carlo simulation encounters difficulties when exploring the topological sector. When we move towards finer lattices the topological quantity such as topological charge freezes. This is due to the fact that the HMC sampling gets restricted to stay in a topological sector for longer run-time. It implies a larger integrated autocorrelation time for topological quantity which is also known as topological freezing. Numerous efforts have been made to mitigate the impact of critical slowing down or topological freezing in statistical systems and lattice QFT [120–122]. However, in lattice gauge theory, it still remains a challenging task.

uring our previous discussions, we introduced a technique for sampling lattice configurations by employing both a C-NF model and a C-GAN [117, 123]. We have shown that the C-NF [117] trained in the non-critical region can produce samples for parameter values in the critical region.

In this chapter, we discuss an equivariant conditional flow model for sampling $U(1)$ gauge theory in 2 dimensions [123]. Our goal is to generate lattice configurations from the distribution for a given β as

$$p(U_\mu(n)|\beta) = \frac{1}{Z} e^{-S(U,\beta)}, \tag{6.1}$$

where $U_\mu(n)$ denotes the $U(1)$ Link variable field, β denotes the lattice action parameter, and Z is the partition function defined as

$$Z = \sum_U e^{-S(U,\beta)}. \tag{6.2}$$

We partition the action parameter into two sets β_L and β_S based on the integrated autocorrelation time (τ_{int}) of topological charge. β_L corresponds to large β values where topological freezing is dominant in HMC simulations (shown in Figure 6.1a) and β_S corresponds to smaller values of β where the τ_{int} is small, and hence the samples fluctuate among different topological sectors as shown in Figure 6.1b. Due to the lower autocorrelation time, the HMC simulation cost is minimal in the β_S set. Hence we train a C-NF model $\tilde{p}(U_\beta)$ with HMC samples from $p(U, \beta_S)$.

We train the C-NF model to be a generalised model over β parameters. The model is then extrapolated to larger values of β, i.e. in the β_L set to generate different ensembles of lattice configurations. However, the extrapolated model may not provide samples from the true target distribution. But the exactness can be guaranteed by using the MH algorithm. So, after training, we use the extrapolated model $\tilde{p}(U(\beta_L))$ at large β values as a proposal for constructing a Markov Chain via an independent MH algorithm[60]. The quality of the extrapolated model may deteriorate as we move further away from the training region, particularly for distant β values. In such cases, we can employ a re-trainable method to sample at those β values. This re-trainable method utilizes the model's own samples at intermediate β value to enhance the generalization of the C-NF model, enabling better sampling capabilities at points that are far away from the training set.

6.1 $U(1)$ Gauge Theory

The lattice action for $U(1)$ gauge theory in $1+1$ dimensions can be written as

$$S(U) = -\beta \sum_{n,\mu<\nu} Re[U_{\mu\nu}(n)], \tag{6.3}$$

where, the plaquette $U_{\mu\nu}$ is defined as

$$U_{\mu\nu} = U_\mu(n) U_\nu(n+\hat{\mu}) U_\mu(n+\hat{\nu}) U_\nu(n). \tag{6.4}$$

$U_\mu(n)$ can be written in terms of angular variable as $U_\mu(n) = e^{i\theta_\mu(n)}$. The action is symmetric under the transformation

$$U_\mu(n) \longrightarrow e^{i\alpha(n)} U_\mu(n) e^{-i\alpha(n+\mu)}. \tag{6.5}$$

In terms of angular variables, the plaquette becomes

$$U_{\mu\nu}(n) = e^{i[\theta_\mu(n)+\theta_\nu(n+\hat{\mu})-\theta_\nu(n+\hat{\nu})-\theta_\nu(n)]}, \tag{6.6}$$

$$= e^{i\theta_{\mu\nu}}(n). \tag{6.7}$$

So, the action can be written as

$$S(U) = -\beta \sum_n \cos\theta_{\mu\nu}(n), \tag{6.8}$$

which is symmetric under

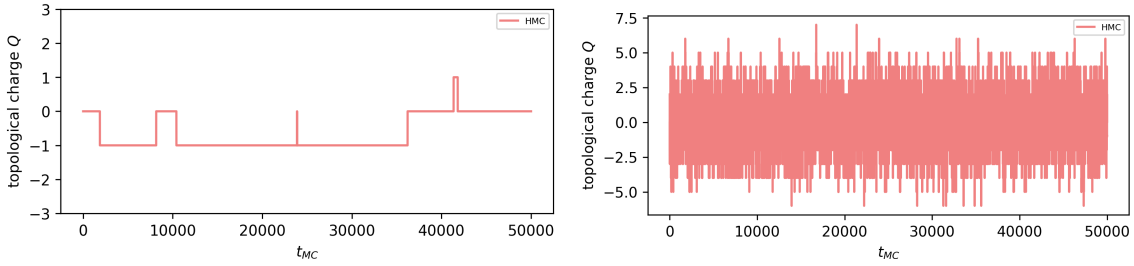
$$\theta_\mu(n) \longrightarrow \alpha(n) + \theta_\mu(n) - \alpha(n+\hat{\mu}), \tag{6.9}$$

$$\text{where} \quad \alpha(n) \in R$$

The action in Equation (6.8) is used in Equation (6.1) for sampling in both HMC and C-NF models. The observable we calculate here is the topological charge defined as

$$Q = \frac{1}{2\pi} \sum_n arg[U_{\mu\nu}(n)], \tag{6.10}$$

$$\text{where,} \quad arg[(U_{\mu\nu}] \in [-\pi, \pi]). \tag{6.11}$$



(A) $\beta = 7$ (B) $\beta = 3.5$

FIGURE 6.1: The fluctuation of topological charge for $50k$ Markov steps simulated by HMC at a) $\beta = 7$ and b) $\beta = 3.5$.

6.2 C-NF Model employing Equivariant Flow

In a flow-based model, starting with a gauge symmetric prior distribution does not guarantee that the resulting output distribution will also be gauge symmetric. The flow transformation must be equivariant for the output distribution to maintain gauge symmetry, meaning it should commute with the gauge transformation. Implementing such equivariance in a flow-based model is non-trivial. However, if the flow transformation only affects the gauge invariant quantity, the abovementioned condition is easily satisfied. In such cases, the flow-based model can maintain gauge symmetry by exclusively transforming the gauge invariant component. In $U(1)$ gauge theory, the link variable $U_\mu(n)$ is not itself gauge invariant. For such construction, one has to find a one-to-one correspondence between the gauge invariant quantity and the link variable. For example, plaquette variable $P_{\mu\nu}(n)$ is one such gauge invariant quantity which can be used to construct equivariant flow. We use the equivariant flow construction as in [62]. In 2d, the link variable is the tensor of shape $(2, L, L)$, and the plaquette variable is the tensor of shape (L, L). The update in $P_{\mu\nu}$ can be translated into an update into a single U_μ using a suitable mask pattern. Note that a single update of a link variable will update two plaquettes. Therefore, in a coupling layer, we need three subclasses of input plaquette, which can be obtained by a suitable mask: active, passive and frozen. The active variables ($P^a_{\mu\nu}$) are transformed by the coupling layer, the frozen variable ($P^f_{\mu\nu}$) are not transformed by this coupling layer and the passive variables ($P^{ps}_{\mu\nu}$) are those which gets transformed indirectly due to a link update. We choose mask such that the link variable $U_{\mu\nu}$ update is sparse and hence scalable to larger lattice sizes. We use 32 coupling layers in the C-NF model.

The link update in a coupling layer happens in two steps; in the first step, we take care of translating $P_{\mu\nu}$ to U_μ and the 2nd step corresponds to the actual flow in the $P_{\mu\nu}(n)$ variable. For the 2nd part, we have used *rational quadratic neural spline flow* (RQ-NSF) and Non-Compact Projection (NCP) flow independently. In the RQ-NSF, we use a fully connected Layer which learns the weight and biases of the spline function, i.e. the final layer is used to construct the knots and derivative of RQ-NSF flow. In NCP flow, we use fully convolutional layers to transform the frozen variables of the coupling layers. Since we use a coupling layer in both cases, the conditional parameter can be concatenated to the frozen variable and passed through the neural network, which is evaluated in the forward direction only.

6.3 Numerical Experiments and Results

In this section, we present the specifics of our numerical experiment and the resulting outcomes. We outline the training and sampling process and the dataset preparation used to train the C-NF model.

6.3.1 Training Dataset

To train the C-NF model, we generate 10 different ensembles of lattice configurations, each corresponding to a different value β, $\beta_S : \{1.0, 1.5, 1.8, 2, 2.2, 2.5, 2.8, 3, 3.2, 3.5\}$. The largest value of β is 3.5, where integrated autocorrelation time is ≈ 23.45. The behaviour of topological charge is shown in the Figure 6.1b for $\beta = 3.5$. To prepare the ensembles, we employ HMC simulation and adjust the HMC parameter for each β value to achieve an acceptance rate of approximately $\approx 85\%$. We have used a non-uniform ensemble size for each β. For the largest value of β, we generate 15k samples, and as we move to lower β, we reduce ensemble size by 500 samples. We perform all the numerical experiments on a 16×16 lattice.

6.3.2 Training and Sampling

In the C-NF model, the input lattice configurations are tensors with a $(2, 16, 16)$ shape. These tensors are concatenated with their corresponding ensemble label or conditional parameter, denoted as $\beta\mathbb{1}$. The tensor $\mathbb{1}$ has the same shape as the lattice configurations, $(2, 16, 16)$, with all entries set to identity values. This means that all masking patterns will be applied to the condition tensor similarly to the lattice configurations. During training, we randomly select a batch from any value of β_S. We use a batch size of 512 to calculate the gradient at each iteration, and the model weights are updated after every ten iterations. The C-NF model is trained using forward KL divergence and an Adam optimizer with an initial learning rate of 0.001. Increasing the Effective Sample Size (ESS) of the C-NF model can be challenging and may reach a plateau during the training. We incorporate a decay of learning rate of 0.5 at intervals of $25k$ training iterations. We have found this approach effective in increasing the ESS for the C-NF model. It's important to note that the C-NF model may exhibit overfitting and perform well only for the training β values. To overcome this, we condition only the coupling layers with index $n \times k$, where $n = 0, 1, ...$ and $k = 1, 5, 9$ allow for better generalization and performance on unseen β values. We stop the training when the ESS reach above 30% and the increment is less than

3% for the next 5k consecutive iterations. Note that one can achieve a higher ESS with further training or a more optimized architecture. However, our objective is to assess the sample quality as we extrapolate for large β values. While training, we also monitor the acceptance rate periodically after every $10k$ iteration from the MH algorithm. At the end of the training, we achieve an acceptance rate of approximately 65% within the training region.

After training, we extrapolate the model for large β values, $\beta_L = \{5.5, 6, 6.5, 7, 7.5\}$. Using the extrapolated model, we generate proposals for the Metropolis-Hastings (MH) step to construct a Markov chain. For each value of β, we generate an ensemble of 10^6 lattice configurations. Once training is over, the flow-based model enables us to generate such large ensembles without any significant challenges efficiently. During the estimation of observables on the ensemble, we leave 20.

6.3.3 Results

We calculate different observables on the ensemble generated from the C-NF model. In Figure 6.2, we have shown the integration autocorrelation time for HMC and C-NF model calculated for a topological charge. For HMC simulation, the effect of topological freezing increases rapidly as we move towards larger β. Since the generation in the C-NF model is inherently uncorrelated, the autocorrelation depends only on the acceptance rate in the MH step. There is a huge gain as we move towards larger β where we have trained the model. In the extrapolated region, the acceptance rate of the MH step is almost

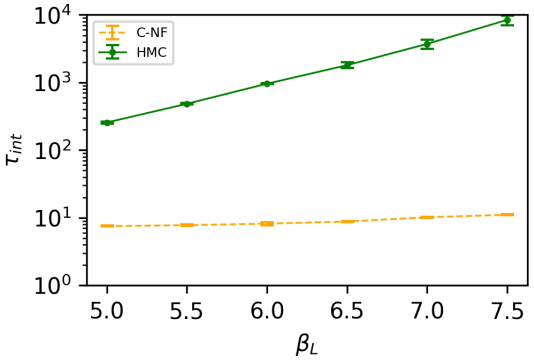


FIGURE 6.2: Integrated autocorrelation time calculated from the C-NF model and HMC simulation is compared. The solid (green) line represents τ_{int} in HMC simulation, and the dashed (orange) line represents τ_{int} in C-NF.

constant. This implies that our model has learned the conditional distribution very well. In Figure 6.3, we show that the acceptance rate is between 60% to 50% for the extrapolated

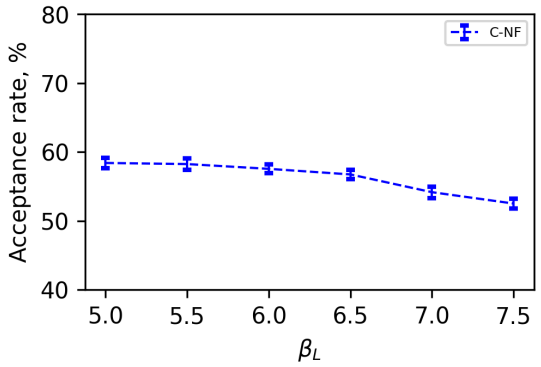


FIGURE 6.3: Acceptance rate calculated from the extrapolated C-NF model for different β values. The acceptance rate has a very small decline over a wide range of non-training regions of β

parameter values in β_L. This allows one to store a single model and generate ensembles at multiple β values.

Figure 6.4 shows the fluctuation of topological charge with the MCMC time. We see clear freezing in the HMC Markov chain. In comparison, the C-NF model has reduced it significantly. This is a major gain one can obtain from a flow-based model.

6.3.4 Re-training Method for Distant β Values

If we want to extrapolate the model far away from the training region, the acceptance rate may decrease. This is due to the lack of good generalization for the conditional model over a wide range of β values. For example, if we want to generate a sample for $\beta = 9$ then the MH acceptance rate drops to $\sim 30\%$. We address this issue by utilizing samples from non-training regions of β values. For extrapolation at $\beta = 9$, first, we generate samples from the C-NF model for an intermediate β, say $\beta = 6$, where the acceptance rate is $\sim 55\%$. We add these newly generated samples and re-train the C-NF model using Forward KL for 10k iterations along with the previously used samples. We reduce 50% samples for each $\beta \in \beta_S$. This will improve the generalization of the C-NF model for larger β values. We observed an increase in acceptance rate with the re-training up to $\sim 40\%$ for $\beta = 9$.

6.4 Summary

This Chapter discussed the difficulties of MCMC sampling of topological quantities in a Lattice Gauge Theory. Estimation of topological charge in MCMC $U(1)$ becomes severe

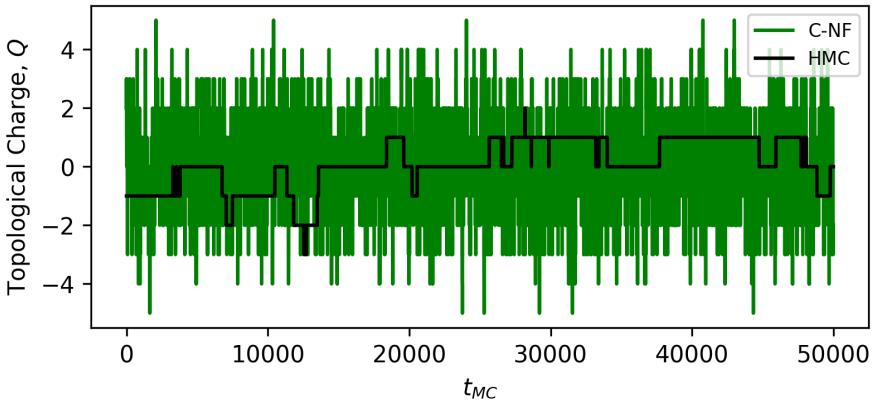


FIGURE 6.4: Topological freezing is shown for $50k$ Markov chain in both HMC simulation and C-NF model. We observe a significant gain from the C-NF model in reducing the effect of topological freezing.

as we approach the theory's continuum limit ($a \to 0$). Therefore, we introduce a Conditional Normalizing Flow (C-NF) model to sample $U(1)$ gauge theory in two dimensions, aiming to mitigate the impact of topological freezing when dealing with smaller values of the $U(1)$ bare coupling. To train the conditional flow model, we use samples generated through the HMC method within a region where the autocorrelation in topological quantities remains minimal. We then extrapolate the model to parameter values where training was not initially conducted. We comprehensively examine the model's performance within this region, aiming to generate uncorrelated samples and significantly reduce instances of topological freezing. Additionally, we propose a re-trainable approach that leverages the model's samples to enhance the generalization capability of the conditional model. This approach enables sampling for coupling values that extend well beyond the initial training region, thereby expanding the model's applicability.

Chapter 7

Summary Conclusion and Future Outlook

This thesis is centred around the advancement of algorithmic techniques in non-perturbative methods within lattice field theory (LFT). The simulation process of LFT is intricate and places substantial computational demands due to the discretization of spacetime and the necessity of extrapolation towards the continuum limit. Efficiently exploring the vast configuration space requires sophisticated algorithms and significant computational resources. Enhancing the efficiency of lattice theory simulations is crucial for advancing our understanding of the strong force, investigating hadron properties, and studying phenomena such as quark-gluon plasma.

The conventional MCMC algorithm for lattice simulation is significantly affected by critical slowing down when adjusting the lattice parameters towards the critical region or continuum limit. At the critical point, the cost of simulating the Hybrid Monte Carlo (HMC) algorithm diverges, particularly in theories like Quantum Chromodynamics (QCD) due to the increasing autocorrelation time. Consequently, generating configurations in LFT near the critical region poses a considerable challenge. In this thesis, the focus lies on constructing and employing Machine Learning (ML) based methods to achieve efficient sampling from a distribution defined by a LFT. We demonstrate how ML-based samplers can effectively generate lattice configurations, overcoming the difficulties encountered by traditional MCMC simulations.

First, we propose a GAN-based approach for sampling the lattice Gross-Neveu model in 2 dimensions. In this approach, we utilize HMC-generated configurations for the Gross-Neveu model away from the critical region and trained a Conditional GAN (C-GAN) to generate lattice configuration near the critical point. With HMC data in a non-critical region, we train the C-GAN model conditioned with parameter λ. For the evaluation of the proposed C-GAN model at the critical region, we compare a few observables on the samples generated from both C-GAN and HMC. We found a good match between the results of HMC and our C-GAN model and also observed that phase transition can be very well reproduced by the generative approach. Since the C-GAN model in the critical region gives correct critical behaviour, we can infer that our generative model is a good interpolator in the critical region. Since C-GAN generates independent configurations, there is no correlation in the samples generated by the C-GAN model, thereby avoiding the critical slowing down problem. In this work, we evaluate our proposed C-GAN model by comparing observables with HMC samples. We could also use our C-GAN model distribution $\hat{p}(\sigma|\lambda)$ as proposal distribution to construct a Markov chain as done in [60]. However, to construct such a Markov chain we must know the proposal density explicitly. which is not available for GANs. Rather in this work, we have accepted all the samples generated by the C-GAN model and thus have a vanishing autocorrelation. In the case of GAN, we can construct a Markov chain in future looking at the recent development regarding density estimation for GAN in the ML community. One such method could be the FlowGAN [124] which explicitly estimates the densities for GAN, where the generator network is replaced by an invertible flow network. Another method could be the Round-trip method [125], which tries to estimate density approximately. Although the problem of critical slowing down is not as severe for the Gross-Neveu (GN) model in 1+1 dimensions, nonetheless GN model serves as a toy model of QCD and building and testing the C-GAN in the GN model establishes its applicability in the lattice formulation of the fermionic system.

It is important to note that while MCMC methods provide theoretical guarantees on the validity of generated samples, GANs do not offer such theoretical guarantees. In the previous case mentioned, the sampling ability of C-GAN in interpolating and extrapolating to critical regions was demonstrated solely through empirical results. To overcome this limitation, we employ a conditional normalizing flow model (C-NF) to efficiently generate proposals for constructing a Markov chain using the independent MH algorithm. We train the C-NF model to learn a conditional distribution over action parameters which enables us to generate proposals at any action parameter of the theory. We apply this method to the scalar lattice ϕ^4 theory. We train the C-NF model on HMC samples in

the non-critical region as earlier. We use the trained model for generating proposals for the construction of a Markov chain. We use an independent Matropolis-Hasting algorithm for this task. Thus the proposed method can generate a Markov chain for any λ in the critical and non-critical regions of the theory. We experimented with both interpolating and extrapolating our model to the critical region. We generate lattice configurations for multiple λ values in the critical region for both cases around 40% acceptance rate. We also compare various observables from our method with that from the HMC simulation and find a good agreement.

The above methods can be extended for lattice simulation of gauge theory. We start with a simple gauge theory such as $U(1)$ gauge theory in 2D. In the simulation of $U(1)$ gauge theory using the MCMC method, the correlation between the samples becomes stronger as lattice spacing decrease i.e., β increases. As β reaches higher values, a phenomenon called topological freezing is observed in the MCMC simulation. To tackle this issue, we have developed a conditional flow-based model (C-NF) specifically designed for sampling the $U(1)$ gauge theory in 2D when β is large. Since the theory has gauge symmetry we have to build a flow model which is equivariant. However, training an equivariant flow with a coupling layer such as a spline is not straightforward and requires special care about the learning rate while training. This model takes β as a conditioning parameter and is trained using ensembles of β values where there are low autocorrelations and minimal effects of topological freezing. The training process continues until the Effective Sample Size (ESS) for the training ensembles reaches approximately 30%.

Once the training is completed, we employ the model to generate samples for larger β values using the Metropolis-Hastings (MH) algorithm. The quality of the extrapolated model depends on how well it has learned a generalized distribution over the β parameter. We achieve an acceptance rate of approximately $50 - 60\%$ in the MH algorithm across a wide range of β values that were not included in the training data. For $\beta = 7.5$, the acceptance rate in the MH algorithm was approximately 52%. However, as β increases further, the acceptance rate gradually decreases. In such cases, we propose a re-trainable method for sampling points that are far beyond the training region. This method utilizes the model's samples at intermediate β values to improve the conditional generation of samples. Furthermore, this flow-based model can also be re-trained for larger lattice sizes as one approaches the continuum limit. One can accomplish this by re-training the model using the reverse Kullback-Leibler (KL) divergence.

The future of lattice field theory research lies in the continued development and application of advanced algorithmic techniques. ML has the potential to play a crucial role in this

regard and may be a game-changer in future QCD simulations. In all the works presented in this thesis, we have tested ML-based approaches on simple lattice theories in (1+1) dimensions. Our ultimate goal is to extend these methods for the simulation of Lattice QCD to make physical predictions. However, before applying this idea to lattice QCD we must make some development in multiple directions. For example, we need to find a way to transfer the ML-based method to the higher dimensional lattice theories. We need to extend our works for non-abelian gauge symmetries like SU(3) gauge theory. Also, we need to scale up the flow-based methods for larger lattices to reduce finite volume effects. These are the potential directions we would like to explore in future and contribute to developments in lattice QCD research.

Appendix A

Machine Learning

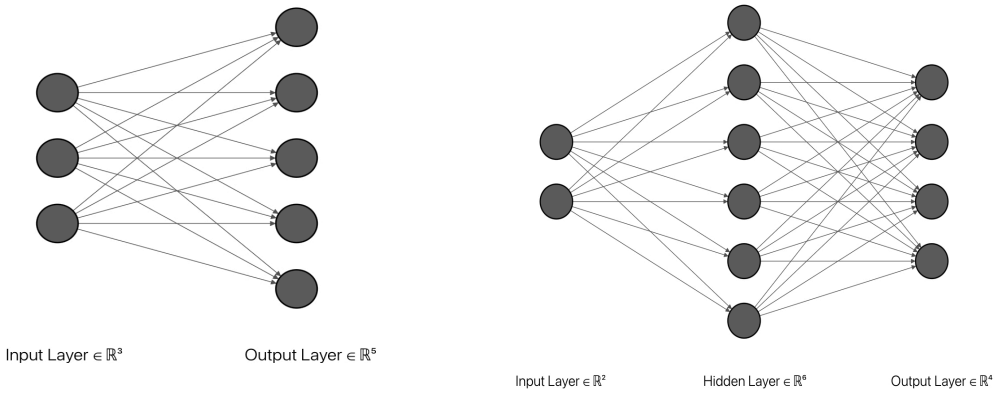
A.1 Few basic terms in ML

Neural Network: A Neural Network is a type of machine learning algorithm that is modelled after the structure and function of the human brain. It is a computational system composed of multiple interconnected processing elements, or nodes, that work together to perform a specific task. Each node in a neural network receives input data x, processes it using a set of weights (w) and biases (b), and then generates an output signal $y = f(wx+b)$, which is passed on to the next layer of nodes. Where f is a nonlinear function introduced to learn more complicated mapping. The weights and biases are adjusted during training to optimize the network's performance.

A feedforward neural network is a neural network where information flows in one direction, from the input layer to the output layer, without any loops or feedback connections. The input layer of a feedforward neural network receives the initial data, which is then passed through one or more hidden layers of interconnected nodes, as shown in Figure A.1. Each node in the hidden layers receives inputs from the previous layer, applies an activation function, and outputs a signal to the next layer.

The output layer of the feedforward neural network produces the final result based on the processed input. During training, the network weights and biases are adjusted using backpropagation, a method for calculating the gradient of the loss function with respect to the network parameters.

Fully Connected Neural Network (FCNN): Fully Connected Neural Network, also known as a dense neural network, is a type of neural network in which every neuron in



(A) Single layer Neural Network (B) Neural Network with hidden layers

FIGURE A.1

one layer is connected to every neuron in the following layer. This means that all the inputs from the previous layer are connected to every neuron in the next layer, and each connection has a weight associated with it. The output of each neuron in the next layer is then calculated using an activation function.

Activation functions: An activation function is a mathematical function that is applied to a neural network node's output. It adds nonlinearity to the network, allowing it to learn complicated patterns in input. Activation functions are typically applied following the computation of the weighted sum of inputs and biases in a node. The result is then sent to the activation function, which transforms the input into a new output value. The following are some typical activation functions used in neural networks:

1. Sigmoid function: The sigmoid function is a common activation function that maps any input value to a value between 0 and 1.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{A.1}$$

   It is often used in the output layer of a binary classification problem.

2. Tanh function: The hyperbolic tangent function, or tanh, is similar to the sigmoid function but maps input values to a range between -1 and 1.

$$\sigma(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{A.2}$$

   It is often used in the hidden layers of neural networks.

3. Rectified Linear Unit (ReLU): ReLU is a simple activation function that returns the input if positive and 0 otherwise.

$$\sigma(x) = \max(0, x) \tag{A.3}$$

It has become one of the most popular activation functions in recent years due to its simplicity and effectiveness.

4. Softmax function: The softmax function is a popular activation function used in the output layer of a neural network to perform multi-class classification. It normalizes the output values to a probability distribution over the classes.

$$\sigma(x_i) = \frac{e_i^x}{\sum_{j=1}^{N} x_j} \tag{A.4}$$

There are many other activation functions that can be used in neural networks, and the choice of activation function can have a significant impact on the performance of the network. Choosing an appropriate activation function for the problem being solved is important.

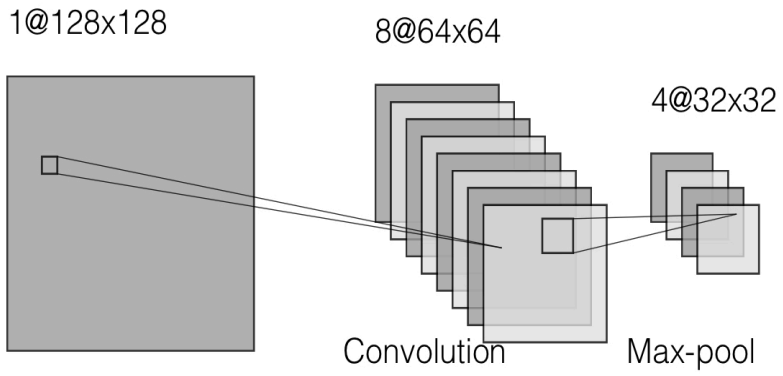


FIGURE A.2: Convolution applied on the image of dimension 128×128 image. After Convolution, the image size changes to 64×64 with eight channels which are then reduced to 32×32 using a max-pooling layer.

Convolutional Neural Networks (CNN): The Fully connected neural networks can be used for a wide range of tasks, including classification, regression, and image recognition. However, as the number of neurons and layers in the network increases, the number of weights and connections can become very large, making training and optimization more difficult. Alternative network architectures, such as Convolutional Neural Networks (CNN)

and Recurrent Neural Networks, have been developed to address this issue. The architecture of a CNN is designed to process data with a grid-like topology, such as an image. It consists of a series of layers, including convolutional layers, pooling layers etc. CNNs are designed to automatically detect and extract features from images by applying filters (kernels or weights) to the input image. The filters are convolved over the image, resulting in feature maps highlighting different aspects of the input image.

A.2 Generative models

A.2.1 Variational autoencoders:

Variational autoencoders (VAEs) [126]are based on the concept of an autoencoder, which consists of two parts: an encoder network that compresses the input data into a lower-dimensional representation and a decoder network that reconstructs the original data from the compressed representation. In a VAE, the encoder network maps the input data to a probability distribution over a lower-dimensional space rather than a specific point. This distribution is typically a multivariate Gaussian, with a mean and variance that are learned during training. The decoder network then takes a sample from this distribution and uses it to reconstruct the original data. One of the advantages of VAEs is that they can generate new data that is different from the training data while preserving the underlying structure of the input data. This makes them useful for data augmentation and image synthesis. However, VAEs can suffer from mode collapse, where the model generates repetitive or unrealistic samples.

A.2.2 Boltzmann Machines:

Boltzmann Machines[127] are generative models consisting of a network of stochastic binary units connected to each other. These models use a Markov Chain Monte Carlo (MCMC) algorithm to generate new data points by sampling from the probability distribution defined by the Boltzmann machine. Boltzmann Machines have been used in various applications, including image recognition, speech recognition, and protein folding. The major difficulty with Boltzmann machines is the computational complexity involved in learning their parameters and making inferences from them.

GAN architecture - an example: The Neural Network architecture of GAN depends on the problem or specifically on the distribution we want to sample. For instance, if we

have a training ensemble of lattice configurations from a generic lattice field theory that is 32×32 in size, then the generator should generate lattices that are of size 32×32. Consider starting the generator with a random vector of size (100). This is then passed to a dense layer and resized to 4×4 with 128 colour channels which will be the input to a CCN layer. We can employ several CNN layers to upsample it to the desired output sample size. After applying each CNN layer, we employed periodic padding. We can use three such CNN layers (Con2DTranspose) with periodic padding, as shown in the Figure A.3. The last convolutional layer does not change the shape of its input. The dimension of the colour channel may change in the intermediate layers. The exact formula which relates the input to an out shape of the convolutional layer can be found in the PyTorch or TensorFlow website `https://www.tensorflow.org/api_docs/python/tf/nn/convolution`. In the

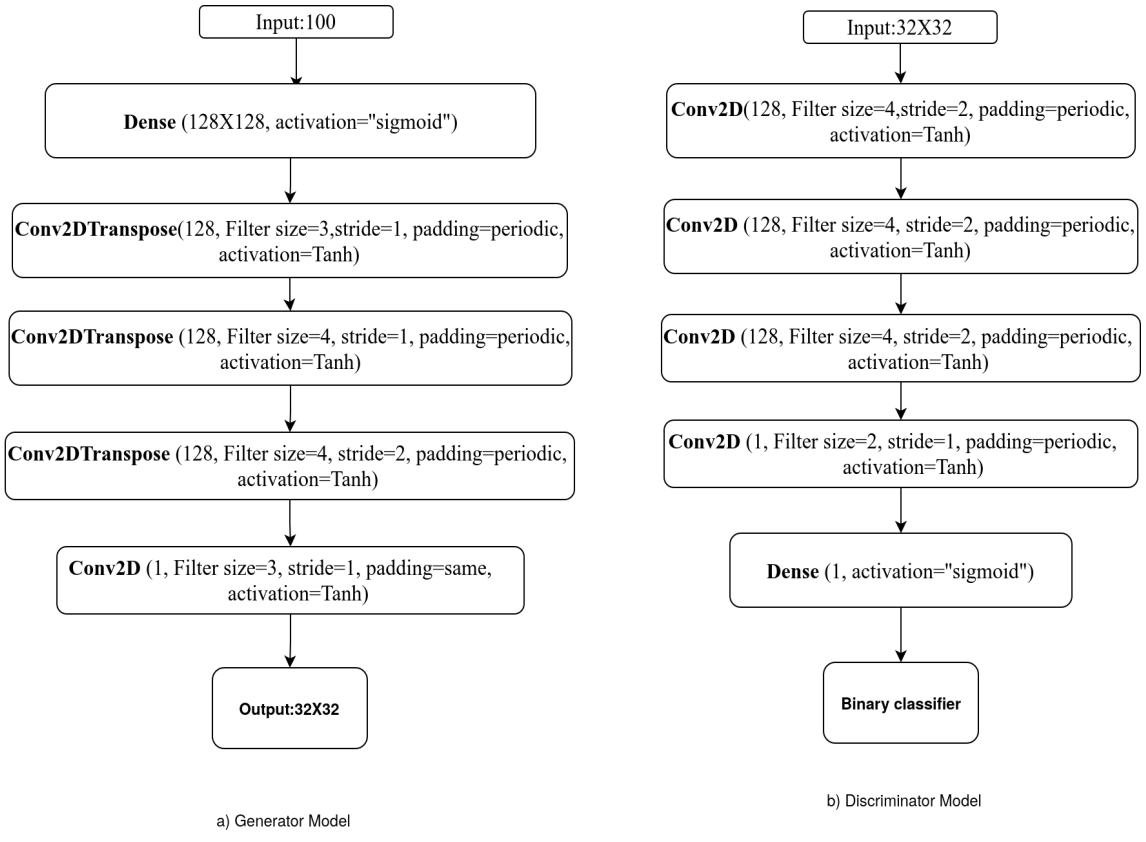


FIGURE A.3: The architecture of the Generator where one dense layer and three convolution layers for up-sampling are used. We use a final convolution layer keeping the output shape the same as the input. The final output shape is (32×32).

discriminator Network, inputs are 32×32 lattices. We need to downsample the lattices, and therefore, four convolutional layers (Conv2D) can be used, followed by a dense layer. Since the discriminator is a binary classifier, we have used a Sigmoid function in the last layer.

A.2.3 Auto-regressive Flows

Auto-regressive models[128, 129] are a class of generative models that generate new data by predicting the next value in a sequence given the previous values. These models are commonly used in language and speech applications, where the sequence represents a sequence of words or phonemes. The most popular auto-regressive model is the GPT series from OpenAI. The GPT model has achieved state-of-the-art results in natural language processing tasks such as language modelling, text generation, and machine translation. Here, we will discuss Auto-regressive models in the context of Normalizing flow.

We can construct an auto-regressive flow model of $p_x(x)$ by decomposing it into a product of one component conditional as

$$p_x(x) = \prod_{i=1}^{D} p_x(x_i|x_{<i}). \tag{A.5}$$

Auto-regressive model sample approximately according to $p(x)$ by conditionally sampling components x_i from approximations to the conditional distributions. This can be done by the composition of flow layers transforming on each component. The transformation can be written as

$$y_i = h(x_i; \Theta_i(x_{1:i-1})), \tag{A.6}$$

here, $x_{1:i} = (x_1, ..., x_i.)$. The function Θ_i is called conditioner, and this can be implemented by using affine transformations

$$y_i = \sigma_i(x_{1:i})x_i + \mu_i(x_{1:i}). \tag{A.7}$$

Note that we do not need the inverse of σ_i or μ_i to find the inverse of the conditioner. The Jacobian matrix of the autoregressive model is triangular and the determinant is just the product of its diagonal entries. Although autoregressive models are quite popular in other areas of machine learning for lattice field theory it is not suitable for sampling purpose as autoregressive flows breaks the locality and translation invariance of the lattice. There we have to look for other flow methods which suit lattice simulation.

A.3 C-NF architecture for Scalar field theory

We construct a C-NF model by an alternate combination of affine layers as shown in Figure 5.1. One such affine layer is shown here. Parameters of the C-NF architecture for different lattice sizes are also listed. Note that we have not conditioned λ to each coupling layer as the model over-fit to the trained λ values. We use a gap of $0,\ 0,\ 4$ and 8 in 6×6, 8×8, 12×12 and 16×16 lattices respectively.

lattice sizes	No. of Affine Layers	$F1, F2, F3$
6×6	8	$16, 16, 2$
8×8	8	$32, 32, 2$
12×12	16	$32, 32, 2$
16×16	24	$16, 16, 2$

TABLE A.1: Parameters of the C-NF architecture for different lattice sizes. $F1, F2$, and $F3$ are as defined in fig. 5.2.

Appendix B

Lattice Field Theory

B.1 Path Integral Formalism:

The path integral in Quantum Mechanics (QM) is based on the principle of superposition, which states that the probability amplitude for a particle to go from one state to another is the sum (or integral) of the probability amplitudes for all possible paths between those states.

The matrix element between two position eigenstates can be written in terms of path integral as

$$\begin{aligned} < x_i, t_i | x_f, t_f \Big\rangle =& \Big\langle x_i | e^{-iH(t_i - t_f)} | x_f \Big\rangle \\ & \propto \int \mathcal{D}x \; e^{\{i \int_{t_f}^{t} dt \mathcal{L}(x, \dot{x})\}}. \end{aligned} \tag{B.1}$$

where, $|x, t >= e^{iH(t-t')}|x_i >$ and $|x', t' = e^{iHt'}|x_f >$.

In the context of interacting quantum fields, the main quantities to calculate are the vacuum expectation values (VEVs) of time-ordered products of field operators, such as the Feynman propagator;

$D_F(x-y) = \Big\langle 0|T[\hat{\phi}(x)...\hat{\phi}(y)]|0 \Big\rangle$.
In QM the analogical quantity is the ; $< x_f, t_f | T(\hat{x}(t_1)\hat{x}(t_2))|x_i, t_i \Big\rangle$.
The time-ordered product of the position operator can be written as

$$< x_f, t_f | T(\hat{x}(t_1)\hat{x}(t_2))|x_i, t_i \Big\rangle \propto \int \mathcal{D}x \; x(t_1)x(t_2) \exp[\; i \int_{t_f}^{t_i} dt \mathcal{L}(x, \dot{x})]. \tag{B.2}$$

where, $\mathcal{L}$ is the Lagrangian of the system.

This can easily generalize to a quantum field theory. Let us consider $\phi(x)$ represents a generic field. If $\phi(t_i)$ and $\phi(t_f)$ are field at time t_f and t_i, then we can the Green's function

$$\Big\langle \phi(t_f)|T(\hat{O}_1...\hat{O}_N)|\phi(t_i)\Big\rangle = \int \mathcal{D}\phi O_1[\phi]...O_N[\phi]e^{iS[\phi]}, \tag{B.3}$$

where the action can be written as an integral over the Lagrangian density,

$$S[\phi] = \int_i^f dx^D \mathcal{L}(\phi(x), \partial_\mu\phi(x)). \tag{B.4}$$

In QFT, we are more interested in the time order correlator than the transformation amplitude between certain field configurations i.e. the object $\Big\langle 0|T[\hat{O}_1...\hat{O}_N]|0\Big\rangle$.

A generic state such as $\phi(t)$ is reasonable to expect that it would have some overlap with the ground state $|0$. if we allow the state to evolve over time while introducing a small imaginary component to the time, the state will gradually decay towards its lowest-energy component, which corresponds to the ground state. Taking the limit $t_{f,i} \to \pm\infty(1 - i\epsilon)$ and the final expression of the time-ordered correlator as

$$\Big\langle 0|T[\hat{O}_1...\hat{O}_N]|0\Big\rangle = \frac{\int \mathcal{D}\phi O_1[\phi], ..., O_N[\phi]e^{iS(\phi)}}{Z}, \tag{B.5}$$

where, $\quad Z = \int \mathcal{D}\phi e^{iS(\phi)}$.

B.1.1 Discrete derivatives on lattice

The partial derivative on the lattice can be defined in different ways namely forward, backward derivative and symmetric derivative. The forward and backward derivatives are defined as

$$\Delta^F_\mu\phi(x) = \frac{1}{a}\big(\phi(x+\hat{\mu}) - \phi(x)\big),$$
$$\Delta^B_\mu\phi(x) = \frac{1}{a}\big(\phi(x) - \phi(x-\hat{\mu})\big).$$

The most popular derivative on the lattice is the symmetric derivative

$$\Delta^{sym}_\mu\phi(x) = \frac{\big(\phi(x+\hat{\mu}) - \phi(x-\hat{\mu})\big)}{2a}. \tag{B.6}$$

The second-order derivative can be written as

$$\Box\phi(x) = \frac{\big(\phi(x+\hat{\mu}) + \phi(x-\hat{\mu}) - 2\phi(x)\big)}{a^2}. \tag{B.7}$$

In a gauge theory with fermions $\psi(n)$ we need gauge covariant derivative. The forward and backward derivatives become,

$$\Delta^F_\mu \psi(n) = \frac{1}{a}\big(U_\mu(n)\psi(n+\hat{\mu}) - \psi(n)\big),$$
$$\Delta^B_\mu \psi(n) = \frac{1}{a}\big(\psi(n) - U^\dagger(n-\hat{\mu})\psi(n-\hat{\mu})\big).$$

And the symmetric derivative is

$$\Delta^{sym}_\mu \psi(x) = \frac{U_\mu(n)\big(\psi(x+\hat{\mu}) - U^\dagger_\mu(n)\psi(x-\hat{\mu})\big)}{2a}. \tag{B.8}$$

B.2 QED action on lattice

The QED action on a lattice in the Wilson fermion formalism can be written as

$$\begin{aligned} S_{QED}[U,\psi,\bar{\psi}] &= S_G + S_F, \\ &= \frac{1}{e^2}\sum_{n,\mu<\nu}\Big[1 - \frac{1}{2}(U_{\mu\nu}(n) + U^\dagger_{\mu\nu}(n))] + (M+4r)\sum_n \bar{\psi}(n)\psi(n) \\ &- \frac{1}{2}\sum_{n,\mu}[\bar{\psi}(n)(r-\gamma_\mu)U_\mu(n)\psi(n+\hat{\mu}) \\ &+ \bar{\psi}(n+\hat{\mu})(r+\gamma_\mu)U^\dagger_\mu(n)\psi(n)]. \end{aligned} \tag{B.9}$$

The action has gauge symmetry under a local gauge transformation G as follows

$$\begin{aligned} &\psi(n) \to G(n)\psi, \\ &\bar{\psi}(n) \to \bar{\psi}(n)G^{-1}(n), \\ &U_{n,n+\hat{\mu}} \to G(n)U_{n,n+\hat{\mu}}G^{-1}(n+\hat{\mu}), \\ &U_{n+\hat{\mu},n} \to G(n+\hat{\mu})U_{n+\hat{\mu},n}G^{-1}(n). \end{aligned}$$

where, $U_{n,n+\hat{\mu}} = e^{i\theta_\mu(n)}$ and $\theta \in [0, 2\pi]$.

Bibliography

[1] G. Zweig, "Origins of the quark model," Proceedings of the Fourth International Conference on Baryon Resonances, 01 1980.

[2] V. E. B. *et. al.*, "Observation of a Hyperon with Strangeness Minus Three," Phys.Rev.Lett, vol. 12, pp. 204–206, Feb. 1964.

[3] D. J. Gross and F. Wilczek, "Ultraviolet behavior of non-abelian gauge theories," Phys. Rev. Lett., vol. 30, pp. 1343–1346, Jun 1973.

[4] H. D. Politzer, "Reliable Perturbative Results for Strong Interactions?," Phys. Rev. Lett., vol. 30, pp. 1346–1349, 1973.

[5] K. G. Wilson, "Confinement of quarks," Phys. Rev. D, vol. 10, pp. 2445–2459, Oct 1974.

[6] F. Karsch, "Lattice qcd at high temperature and density," in Lectures on quark matter, pp. 209–249, Springer, 2002.

[7] E. Laermann and O. Philipsen, "Lattice qcd at finite temperature," Annual Review of Nuclear and Particle Science, vol. 53, no. 1, pp. 163–198, 2003.

[8] E. Chang, W. Detmold, K. Orginos, A. Parreno, M. J. Savage, B. C. Tiburzi, S. R. Beane, N. Collaboration, et al., "Magnetic structure of light nuclei from lattice qcd," Physical Review D, vol. 92, no. 11, p. 114502, 2015.

[9] F. Winter, W. Detmold, A. S. Gambhir, K. Orginos, M. J. Savage, P. E. Shanahan, M. L. Wagman, N. Collaboration, et al., "First lattice qcd study of the gluonic structure of light nuclei," Physical Review D, vol. 96, no. 9, p. 094512, 2017.

[10] Abi et al., "Measurement of the positive muon anomalous magnetic moment to 0.46 ppm," Phys. Rev. Lett., vol. 126, p. 141801, Apr 2021.

[11] T.Aoyama et al., "The anomalous magnetic moment of the muon in the standard model," Physics Reports, vol. 887, pp. 1–166, 2020. The anomalous magnetic moment of the muon in the Standard Model.

[12] G.Colangelo et al., "Prospects for precise predictions of a_μ in the Standard Model," arXiv e-prints, p. arXiv:2203.15810, Mar. 2022.

[13] P. A. Boyle et al., "A lattice QCD perspective on weak decays of b and c quarks Snowmass 2022 White Paper," tech. rep., 2022. contribution to Snowmass 2021; 19 pages; v2 corrected typo and added references.

[14] T. Blum et al., "Discovering new physics in rare kaon decays," in Snowmass 2021, 3 2022.

[15] U. Wolff, "CRITICAL SLOWING DOWN," Nucl. Phys. B Proc. Suppl., vol. 17, pp. 93–102, 1990.

[16] S. Schaefer, R. Sommer, and F. Virotta, "Critical slowing down and error analysis in lattice QCD simulations," Nucl. Phys. B, vol. 845, pp. 93–119, 2011.

[17] H. Meyer, H. Simma, R. Sommer, M. Della Morte, O. Witzel, and U. Wolff, "Exploring the hmc trajectory-length dependence of autocorrelation times in lattice qcd," Computer Physics Communications, vol. 176, no. 2, pp. 91–97, 2007.

[18] C. Andrieu and J. Thoms, "A tutorial on adaptive mcmc," Statistics and computing, vol. 18, pp. 343–373, 2008.

[19] M. D. Hoffman and A. Gelman, "The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo," 2011.

[20] R. G. Edwards and B. Joó, "The chroma software system for lattice QCD," Nuclear Physics B - Proceedings Supplements, vol. 140, pp. 832–834, mar 2005.

[21] T. Preis, P. Virnau, W. Paul, and J. J. Schneider, "Gpu accelerated monte carlo simulation of the 2d and 3d ising model," Journal of Computational Physics, vol. 228, no. 12, pp. 4468–4477, 2009.

[22] B. Block, P. Virnau, and T. Preis, "Multi-GPU accelerated multi-spin monte carlo simulations of the 2d ising model," Computer Physics Communications, vol. 181, pp. 1549–1556, sep 2010.

[23] M. A. Clark, B. Joó, A. Strelchenko, M. Cheng, A. Gambhir, and R. Brower, "Accelerating lattice qcd multigrid on gpus using fine-grained parallelization," 2016.

[24] M. Clark, R. Babich, K. Barros, R. Brower, and C. Rebbi, "Solving lattice qcd systems of equations using mixed precision solvers on gpus," Computer Physics Communications, vol. 181, no. 9, pp. 1517–1528, 2010.

[25] R. Babich, J. Brannick, R. C. Brower, M. A. Clark, T. A. Manteuffel, S. F. McCormick, J. C. Osborn, and C. Rebbi, "Adaptive multigrid algorithm for the lattice wilson-dirac operator," Physical Review Letters, vol. 105, nov 2010.

[26] R. Babich, M. A. Clark, and B. Joó, "Parallelizing the QUDA library for multi-GPU calculations in lattice quantum chromodynamics," in 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, nov 2010.

[27] K. A. Hawick, A. Leist, and D. P. Playne, "Regular lattice and small-world spin model simulations using cuda and gpus," International Journal of Parallel Programming, vol. 39, pp. 183–201, 2011.

[28] R. H. Swendsen and J.-S. Wang, "Nonuniversal critical dynamics in monte carlo simulations," Phys. Rev. Lett., vol. 58, pp. 86–88, Jan 1987.

[29] J. Hoshen and R. Kopelman, "Percolation and cluster distribution. i. cluster multiple labeling technique and critical concentration algorithm," Phys. Rev. B, vol. 14, pp. 3438–3445, Oct 1976.

[30] U. Wolff, "Monte carlo simulation of a lattice field theory as correlated percolation," Nuclear Physics B, vol. 300, pp. 501–516, 1988.

[31] U. Wolff, "Collective monte carlo updating for spin systems," Phys. Rev. Lett., vol. 62, pp. 361–364, Jan 1989.

[32] W. Bietenholz, A. Pochinsky, and U. J. Wiese, "Meron-cluster simulation of the θ vacuum in the 2d o(3) model," Phys. Rev. Lett., vol. 75, pp. 4524–4527, Dec 1995.

[33] H. G. Evertz, "The loop algorithm," Advances in Physics, vol. 52, no. 1, pp. 1–66, 2003.

[34] N. Prokof'ev and B. Svistunov, "Worm algorithms for classical statistical models," Phys. Rev. Lett., vol. 87, p. 160601, Sep 2001.

[35] H. Neuberger, "Adler's overrelaxation algorithm for goldstone bosons," Phys. Rev. Lett., vol. 59, pp. 1877–1880, Oct 1987.

[36] R. C. Brower and P. Tamayo, "Embedded dynamics for φ^4 theory," Phys. Rev. Lett., vol. 62, pp. 1087–1090, Mar 1989.

[37] M. Hasenbusch, "Improved estimators for a cluster updating of o(n) spin models," Nuclear Physics B, vol. 333, no. 2, pp. 581–592, 1990.

[38] R. Sinclair, "Cluster algorithm for two-dimensional u(1) lattice gauge theory," Phys. Rev. D, vol. 45, pp. 2098–2100, Mar 1992.

[39] L. D. Debbio, H. Panagopoulos, and E. Vicari, "*theta* dependence of su(n) gauge theories," Journal of High Energy Physics, vol. 2002, p. 044, sep 2002.

[40] S. Schaefer, R. Sommer, and F. Virotta, "Investigating the critical slowing down of qcd simulations," 2009.

[41] M. Luscher, "Topology, the wilson flow and the hmc algorithm," 2014.

[42] D. Guest, K. Cranmer, and D. Whiteson, "Deep learning and its application to lhc physics," Annual Review of Nuclear and Particle Science, vol. 68, no. 1, pp. 161–181, 2018.

[43] A. Butter and T. Plehn, "Generative networks for lhc events," 2020.

[44] A. J. Larkoski, I. Moult, and B. Nachman, "Jet substructure at the large hadron collider: A review of recent advances in theory and machine learning," Physics Reports, vol. 841, pp. 1–63, jan 2020.

[45] A. Butter et al., "Machine learning and LHC event generation," SciPost Physics, vol. 14, apr 2023.

[46] B. Nachman, "Anomaly detection for physics analysis and less than supervised learning," 2020.

[47] K. Zhou, G. Endrődi, L.-G. Pang, and H. Stöcker, "Regressive and generative neural networks for scalar field theory," Phys. Rev. D, vol. 100, p. 011501, Jul 2019.

[48] D. Wu, L. Wang, and P. Zhang, "Solving statistical mechanics using variational autoregressive networks," Phys. Rev. Lett., vol. 122, p. 080602, Feb 2019.

[49] J. M. Pawlowski and J. M. Urban, "Reducing autocorrelation times in lattice simulations with generative adversarial networks," Machine Learning: Science and Technology, vol. 1, p. 045011, oct 2020.

[50] K. A. Nicoli, S. Nakajima, N. Strodthoff, W. Samek, K.-R. Müller, and P. Kessel, "Asymptotically unbiased estimation of physical observables with neural samplers," Physical Review E, vol. 101, feb 2020.

[51] J. Carrasquilla, "Machine learning for quantum matter," Advances in Physics: X, vol. 5, p. 1797528, jan 2020.

[52] J. Liu, H. Shen, Y. Qi, Z. Y. Meng, and L. Fu, "Self-learning monte carlo method and cumulative update in fermion systems," Physical Review B, vol. 95, jun 2017.

[53] J. Vielhaben and N. Strodthoff, "Generative neural samplers for the quantum heisenberg chain," Physical Review E, vol. 103, jun 2021.

[54] C. Chen, X. Y. Xu, J. Liu, G. Batrouni, R. Scalettar, and Z. Y. Meng, "Symmetry-enforced self-learning monte carlo method applied to the holstein model," Physical Review B, vol. 98, jul 2018.

[55] G. Torlai and R. G. Melko, "Learning thermodynamics with boltzmann machines," Phys. Rev. B, vol. 94, p. 165134, Oct 2016.

[56] G. Carleo and M. Troyer, "Solving the quantum many-body problem with artificial neural networks," Science, vol. 355, p. 602, 2017.

[57] L. Wang, "Discovering phase transitions with unsupervised learning," Phys. Rev. B, vol. 94, p. 195105, Nov 2016.

[58] P. Zhang, H. Shen, and H. Zhai, "Machine learning topological invariants with neural networks," Phys. Rev. Lett., vol. 120, p. 066401, Feb 2018.

[59] P. E. Shanahan, D. Trewartha, and W. Detmold, "Machine learning action parameters in lattice quantum chromodynamics," Physical Review D, vol. 97, no. 9, p. 094506, 2018.

[60] 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, vol. 100, p. 034515, Aug 2019.

[61] M. S. Albergo, G. Kanwar, S. Racanière, D. J. Rezende, J. M. Urban, D. Boyda, K. Cranmer, D. C. Hackett, and P. E. Shanahan, "Flow-based sampling for fermionic lattice field theories," Phys. Rev. D, vol. 104, no. 11, p. 114507, 2021.

[62] 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," Physical Review Letters, vol. 125, Sep 2020.

[63] M. S. Albergo, D. Boyda, D. C. Hackett, G. Kanwar, K. Cranmer, S. Racanière, D. J. Rezende, and P. E. Shanahan, "Introduction to normalizing flows for lattice field theory," 2021.

[64] M. S. Albergo, D. Boyda, K. Cranmer, D. C. Hackett, G. Kanwar, S. Racanière, D. J. Rezende, F. Romero-López, P. E. Shanahan, and J. M. Urban, "Flow-based sampling in the lattice Schwinger model at criticality," Phys. Rev. D, vol. 106, no. 1, p. 014514, 2022.

[65] D. C. Hackett, C.-C. Hsieh, M. S. Albergo, D. Boyda, J.-W. Chen, K.-F. Chen, K. Cranmer, G. Kanwar, and P. E. Shanahan, "Flow-based sampling for multimodal distributions in lattice field theory," arXiv:2107.00734, 7 2021.

[66] K. A. Nicoli, C. J. Anders, L. Funcke, T. Hartung, K. Jansen, P. Kessel, S. Nakajima, and P. Stornati, "Estimation of thermodynamic observables in lattice field theories with deep generative models," Phys. Rev. Lett., vol. 126, p. 032001, Jan 2021.

[67] O. Philipsen, "The qcd equation of state from the lattice," Progress in Particle and Nuclear Physics, vol. 70, pp. 55–107, 2013.

[68] H. J. Rothe, Lattice Gauge Theories : An Introduction (Fourth Edition), vol. 43. World Scientific Publishing Company, 2012.

[69] C. Gattringer and C. B. Lang, Quantum chromodynamics on the lattice, vol. 788. Berlin: Springer, 2010.

[70] J. Smit, Introduction to quantum fields on a lattice: A robust mate, vol. 15. Cambridge University Press, 1 2011.

[71] I. Montvay and G. Munster, Quantum Fields on a Lattice. Cambridge Monographs on Mathematical Physics, Cambridge University Press, 1994.

[72] R. P. Feynman, "Space-time approach to non-relativistic quantum mechanics," Rev. Mod. Phys., vol. 20, pp. 367–387, Apr 1948.

[73] R. P. Feynman, A. R. Hibbs, and D. F. Styer, Quantum mechanics and path integrals. Courier Corporation, 2010.

[74] K. Symanzik, "Continuum limit and improved action in lattice theories: (i). principles and phi4 theory," Nuclear Physics B, vol. 226, no. 1, pp. 187–204, 1983.

[75] Y. Shamir, "Chiral fermions from lattice boundaries," Nuclear Physics B, vol. 406, pp. 90–106, sep 1993.

[76] R. Narayanan and H. Neuberger, "Chiral fermions on the lattice," Physical Review Letters, vol. 71, pp. 3251–3254, nov 1993.

[77] D. B. Kaplan, "A method for simulating chiral fermions on the lattice," Physics Letters B, vol. 288, pp. 342–347, aug 1992.

[78] A. Borici, "Truncated overlap fermions," arXiv preprint hep-lat/9909057, 1999.

[79] H. Neuberger, "Exactly massless quarks on the lattice," Physics Letters B, vol. 417, no. 1-2, pp. 141–144, 1998.

[80] R. Frezzotti and G. C. Rossi, "Chirally improving wilson fermions 1. o (a) improvement," Journal of High Energy Physics, vol. 2004, no. 08, p. 007, 2004.

[81] S. Capitani, M. Creutz, J. Weber, and H. Wittig, "Renormalization of minimally doubled fermions," JHEP, vol. 09, p. 027, 2010.

[82] A. Borici, "Creutz fermions on an orthogonal lattice," Physical Review D, vol. 78, no. 7, p. 074504, 2008.

[83] M. Creutz, T. Kimura, and T. Misumi, "Aoki Phases in the Lattice Gross-Neveu Model with Flavored Mass terms," Phys. Rev. D, vol. 83, p. 094506, 2011.

[84] J. Goswami, D. Chakrabarti, and S. Basak, "Gross-Neveu model with Borici-Creutz fermion," Phys. Rev. D, vol. 91, no. 1, p. 014507, 2015.

[85] J. Goswami, D. Chakrabarti, and S. Basak, "Mass spectroscopy using Borici–Creutz fermions on 2D lattice," Int. J. Mod. Phys. A, vol. 32, no. 11, p. 1750059, 2017.

[86] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques. Morgan Kaufmann Series in Data Management Systems, Amsterdam: Morgan Kaufmann, 3 ed., 2011.

[87] M. Fatima and M. Pasha, "Survey of machine learning algorithms for disease diagnostic," Journal of Intelligent Learning Systems and Applications, vol. 09, pp. 1–16, 2017.

[88] M. Nilashi, O. Ibrahim, H. Ahmadi, and L. Shahmoradi, "An analytical method for diseases prediction using machine learning techniques," Comput. Chem. Eng., vol. 106, pp. 212–223, 2017.

[89] S. Shen, H. Jiang, and T. Zhang, "Stock market forecasting using machine learning algorithms," Department of Electrical Engineering, Stanford University, Stanford, CA, pp. 1–5, 2012.

[90] M. Umer, M. Awais, and M. Muzammul, "Stock market prediction using machine learning (ml) algorithms," ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal, vol. 8, no. 4, pp. 97–116, 2019.

[91] C. Dvorkin, S. Mishra-Sharma, B. Nord, V. A. Villar, C. Avestruz, K. Bechtol, A. Ćiprijanović, A. J. Connolly, L. H. Garrison, G. Narayan, and F. Villaescusa-Navarro, "Machine learning and cosmology," 2022.

[92] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," Nature, vol. 549, no. 7671, pp. 195–202, 2017.

[93] W. Guan, G. Perdue, A. Pesah, M. Schuld, K. Terashi, S. Vallecorsa, and J.-R. Vlimant, "Quantum machine learning in high energy physics," Machine Learning: Science and Technology, vol. 2, p. 011003, mar 2021.

[94] W. O'Quinn and S. Mao, "Quantum machine learning: Recent advances and outlook," IEEE Wireless Communications, vol. 27, no. 3, pp. 126–131, 2020.

[95] V. Dunjko, J. M. Taylor, and H. J. Briegel, "Quantum-enhanced machine learning," Phys. Rev. Lett., vol. 117, p. 130501, Sep 2016.

[96] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," Contemporary Physics, vol. 56, no. 2, pp. 172–185, 2015.

[97] L. Huang and L. Wang, "Accelerated monte carlo simulations with restricted boltzmann machines," Phys. Rev. B, vol. 95, p. 035105, Jan 2017.

[98] J. Liu, Y. Qi, Z. Y. Meng, and L. Fu, "Self-learning monte carlo method," Phys. Rev. B, vol. 95, p. 041101, Jan 2017.

[99] A. Tanaka and A. Tomiya, "Towards reduction of autocorrelation in hmc by machine learning," 2017.

[100] S. L. Cessie and J. C. V. Houwelingen, "Ridge estimators in logistic regression," Journal of the Royal Statistical Society. Series C (Applied Statistics), vol. 41, no. 1, pp. 191–201, 1992.

[101] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in Proc. 5th Berkeley Symposium on Math., Stat., and Prob, p. 281, 1965.

[102] M. Mohammed, M. B. Khan, and E. B. M. Bashier, Machine learning: algorithms and applications. Crc Press, 2016.

[103] 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," IEEE Transactions on Pattern Analysis & Machine Intelligence, vol. 44, pp. 7327–7347, nov 2022.

[104] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[105] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," 2016.

[106] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing flows: An introduction and review of current methods," IEEE transactions on pattern analysis and machine intelligence, vol. 43, no. 11, pp. 3964–3979, 2020.

[107] A. Hyvärinen and P. Pajunen, "Nonlinear independent component analysis: Existence and uniqueness results," Neural Networks, vol. 12, no. 3, pp. 429–439, 1999.

[108] D. J. Gross and A. Neveu, "Dynamical symmetry breaking in asymptotically free field theories," Phys. Rev. D, vol. 10, pp. 3235–3253, Nov 1974.

[109] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014.

[110] M. Thies and K. Urlichs, "Revised phase diagram of the gross-neveu model," Physical Review D, vol. 67, Jun 2003.

[111] O. Schnetz, M. Thies, and K. Urlichs, "Phase diagram of the gross–neveu model: exact results and condensed matter precursors," Annals of Physics, vol. 314, p. 425–447, Dec 2004.

[112] L. Pannullo, J. Lenz, M. Wagner, B. Wellegehausen, and A. Wipf, "Inhomogeneous phases in the 1+1 dimensional gross–neveu model at finite number of fermion flavors," Acta Physica Polonica B Proceedings Supplement, vol. 13, no. 1, p. 127, 2020.

[113] J. Lenz, L. Pannullo, M. Wagner, B. Wellegehausen, and A. Wipf, "Inhomogeneous phases in the gross-neveu model in 1+1 dimensions at finite number of flavors," Phys. Rev. D, vol. 101, no. 9, p. 094512, 2020.

[114] P. De Forcrand and U. Wenger, "New baryon matter in the lattice gross-neveu model," PoS, vol. LAT2006, p. 152, 2006.

[115] A. Singha, D. Chakrabarti, and V. Arora, "Generative learning for the problem of critical slowing down in lattice Gross-Neveu model," SciPost Phys. Core, vol. 5, p. 052, 2022.

[116] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," 2015.

[117] A. Singha, D. Chakrabarti, and V. Arora, "Conditional normalizing flow for Markov chain Monte Carlo sampling in the critical region of lattice field theory," Phys. Rev. D, vol. 107, no. 1, p. 014512, 2023.

[118] L. Ardizzone, C. Lüth, J. Kruse, C. Rother, and U. Köthe, "Guided image generation with conditional invertible neural networks," CoRR, vol. abs/1907.02392, 2019.

[119] I. Vierhaus, "Simulation of phi 4 theory in the strong coupling expansion beyond the ising limit," Master's thesis, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät I, 2010.

[120] A. Ramos, "Playing with the kinetic term in the HMC," PoS, vol. LATTICE2012, p. 193, 2012.

[121] A. S. Gambhir and K. Orginos, "Improved sampling algorithms in lattice qcd," 2015.

[122] M. G. Endres, R. C. Brower, W. Detmold, K. Orginos, and A. V. Pochinsky, "Multi-scale monte carlo equilibration: Pure yang-mills theory," Physical Review D, vol. 92, Dec 2015.

[123] A. Singha, D. Chakrabarti, and V. Arora, "Sampling u(1) gauge theory using a re-trainable conditional flow-based model," 2023.

[124] A. Grover, M. Dhar, and S. Ermon, "Flow-gan: Bridging implicit and prescribed learning in generative models," CoRR, vol. abs/1705.08868, 2017.

[125] Q. Liu, J. Xu, R. Jiang, and W. H. Wong, "Density estimation using deep generative neural networks," Proceedings of the National Academy of Sciences, vol. 118, no. 15, p. e2101344118, 2021.

[126] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2022.

[127] G. E. Hinton, "A practical guide to training restricted boltzmann machines," Neural Networks: Tricks of the Trade: Second Edition, pp. 599–619, 2012.

[128] M. Germain, K. Gregor, I. Murray, and H. Larochelle, "Made: Masked autoencoder for distribution estimation," 2015.

[129] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in Advances in Neural Information Processing Systems (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds.), vol. 27, Curran Associates, Inc., 2014.

[130] B. Joó, C. Jung, N. H. Christ, W. Detmold, R. G. Edwards, M. Savage, and P. Shanahan, "Status and future perspectives for lattice gauge theory calculations to the exascale and beyond," European Physical Journal A, vol. 55, p. 199, Nov. 2019.

[131] L. Ardizzone, C. Lüth, J. Kruse, C. Rother, and U. Köthe, "Guided image generation with conditional invertible neural networks," 2019.

[132] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," 2019.

[133] B. Durhuus, "ON THE STRUCTURE OF GAUGE INVARIANT CLASSICAL OBSERVABLES IN LATTICE GAUGE THEORIES," 1980.

[134] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," The Journal of Machine Learning Research, vol. 22, no. 1, pp. 2617–2680, 2021.

[135] V. I. Bogachev, A. V. Kolesnikov, and K. Medvedev, "Triangular transformations of measures," Sbornik Mathematics, vol. 196, pp. 309–335, 2005.

[136] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid monte carlo," Physics letters B, vol. 195, no. 2, pp. 216–222, 1987.

[137] R. Stratonovich, "On a method of calculating quantum distribution functions," in Soviet Physics Doklady, vol. 2, p. 416, 1957.

[138] J. Lenz, L. Pannullo, M. Wagner, B. Wellegehausen, and A. Wipf, "Inhomogeneous phases in the gross-neveu model in 1+1 dimensions at finite number of flavors," Physical Review D, vol. 101, May 2020.

[139] C. Gattringer and C. Lang, Quantum chromodynamics on the lattice: an introductory presentation, vol. 788. Springer Science & Business Media, 2009.

[140] A. J. Beekman, L. Rademaker, and J. van Wezel, "An Introduction to Spontaneous Symmetry Breaking," SciPost Phys. Lect. Notes, p. 11, 2019.

[141] J. Lenz, L. Pannullo, M. Wagner, B. Wellegehausen, and A. Wipf, "Inhomogeneous phases in the gross-neveu model in $1 + 1$ dimensions at finite number of flavors," Phys. Rev. D, vol. 101, p. 094512, May 2020.

[142] L. Pannullo, "Master thesis,inhomogeneous phases in the 1 + 1-dimensional gross-neveu model at finite number of fermion flavors," 2020.

[143] J. Singh, M. Scheurer, and V. Arora, "Conditional generative models for sampling and phase transition indication in spin systems," SciPost Physics, vol. 11, Aug 2021.

[144] T. Korzec, F. Knechtli, U. Wolff, and B. Leder, "Monte-Carlo simulation of the chiral Gross-Neveu model," PoS, vol. LAT2005, p. 267, 2006.

[145] J. Danzer and C. Gattringer, "Excited state spectroscopy in the lattice gross-neveu model," PoS, vol. LATTICE2007, p. 092, 2007.

[146] L. Susskind, "Lattice Fermions," Phys. Rev. D, vol. 16, pp. 3031–3039, 1977.

[147] S. Aoki, "New Phase Structure for Lattice QCD with Wilson Fermions," Phys. Rev. D, vol. 30, p. 2653, 1984.

[148] S. Schaefer, R. Sommer, and F. Virotta, "Critical slowing down and error analysis in lattice qcd simulations," Nuclear Physics B, vol. 845, p. 93–119, Apr 2011.

[149] D. Boyda, G. Kanwar, S. Racanière, D. J. Rezende, M. S. Albergo, K. Cranmer, D. C. Hackett, and P. E. Shanahan, "Sampling using su(n) gauge equivariant flows," Physical Review D, vol. 103, Apr 2021.

[150] M. Gabrié, G. M. Rotskoff, and E. Vanden-Eijnden, "Adaptive monte carlo augmented with normalizing flows," 2021.

[151] F. Chollet et al., "Keras conv2d layer." `https://keras.io/api/layers/convolution_layers/convolution2d/`, 2015.

[152] D. Albandea, P. Hernández, A. Ramos, and F. Romero-López, "Topological sampling through windings," Eur. Phys. J. C, vol. 81, no. 10, p. 873, 2021.