Neural Quantum States for Scientific Computing: Applications to Computational Chemistry and Finance

by

Tianchen Zhao

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Applied and Interdisciplinary Mathematics)
in the University of Michigan
2022

Doctoral Committee:

Professor Vikram Gavini, Co-Chair
Professor Shravan Veerapaneni, Co-Chair
Assistant Professor Asaf Cohen
Dr. James Stokes, Flatiron Institute

Tianchen Zhao
ericolon@umich.edu
ORCID iD: 0000-0003-2911-7205

Acknowledgments

I would like to express my special appreciation to my advisor Professor Shravan Veerapaneni, for his help and encouragement during my studies at the University of Michigan. My doctoral experience is a special one, and there were ups and downs over the trajectory of my graduate study. Shravan gives me tremendous support at the time I needed the most and on forth, and has been a reliable source of guidance on both academy and personal development. In particular, his capability of leveraging my strength with his line of research is a great fortune to me. I would like to thank James Stokes for all the insightful discussions; his advice on many aspects of my research has been invaluable. I am extremely grateful to Professor Vikram Gavini, Professor Asaf Cohen, and Dr. James Stokes for agreeing to serve as my committee members and for their detailed feedback.

Preface

This thesis consists of six collaborated projects, including both published papers and to-be-submitted manuscripts. Each project is established into an individual chapter: the first three chapters focus on the fundamental developments of our variational quantum Monte Carlo (VQMC) method, and the next two are its applications to quantum chemistry and financial derivative pricing, respectively. The last chapter is not closely related to VQMC but can be thought of as a general machine learning technique applicable to our context.

1. Natural evolution strategies and variational monte carlo. T. Zhao, G. Carleo, J. Stokes, and S. Veerapaneni. *Machine Learning: Science and Technology*, 2020.
2. Meta variational Monte Carlo. T. Zhao, J. Stokes, O. Knitter, B. Chen, and S. Veerapaneni. *NeurIPS Workshop on Machine Learning and the Physical Sciences*, 2020.
3. Overcoming barriers to scalability in variational quantum monte carlo. T. Zhao, S. De, B. Chen, J. Stokes, and S. Veerapaneni. *The International Conference for High Performance Computing, Networking, Storage, and Analysis*, 2021.
4. Scalable neural quantum states architecture for quantum chemistry. T. Zhao, J. Stokes, S. Veerapaneni. *arXiv preprint*, 2022.
5. Quantum-inspired variational algorithms for partial differential equations: Application to financial derivative pricing. T. Zhao, C. Sun, A. Cohen, J. Stokes, S. Veerapaneni. *arXiv preprint arXiv:2207.10838*, 2022.

Table of Contents

List of Figures

List of Tables

Abstract

The variational quantum Monte Carlo (VQMC) method has received significant attention because of its ability to overcome the curse of dimensionality inherent in many-body quantum systems, by representing the exponentially complex quantum states variationally with machine learning models. We develop novel training strategies to improve the scalability of VQMC, and build parallelization frameworks for solving large-scale problems. The application of our method is extended to quantum chemistry and financial derivative pricing. For quantum chemistry, we build a pre-processing pipeline serving as an interface connecting molecular information and VQMC, and achieve remarkable performance in comparison with the classical approximate methods. On the other hand, we present a simple generalization of VQMC applicable to arbitrary linear PDEs, showcasing the technique in the Black-Scholes equation for pricing European contingent claims dependent on many underlying assets. We also introduce meta-learning and multi-fidelity active learning as exotic components to VQMC, which, under some reasonable assumptions on the problem formulation, can further improve the convergence and the sampling efficiency of our method.

Chapter 1

Introduction

Quantum many-body system describes a vast category of physical problems, in which the repeated interactions between particles create quantum correlations, and the wave function of the system is a complicated object holding a large amount of information, which makes exact calculations about the system impractical. The variational quantum Monte Carlo (VQMC) method received significant attention in the recent past because of its ability to overcome the curse of dimensionality inherent in quantum many-body systems, by representing the quantum states variationally with machine learning models. In particular, VQMC transforms the eigenvalue problem into a stochastic optimization problem, and the exact ground-state wavefunction can be found as the solution to the optimization problem. It is in general impossible to compute the optimization objective exactly, since it involves integrals over a high-dimensional space. Instead, this population quantity is approximated with statistical averages over finite samples from the probability distribution inferred by the wavefunction. The idea of utilizing neural-network quantum states to overcome the curse of dimensionality in high-dimensional VQMC simulations was first introduced by Carleo and Troyer [24], who concentrated on restricted Boltzmann machines (RBMs) applied to two-dimensional quantum spin models. Gomes et al. [47] shows that techniques from quantum VQMC literature can be adapted for approximately solving combinatorial optimization problems.

Dispite the recent surge of interest in VQMC-related methods applying to quantum many-body problems, researchers encountered limitations when working with their fields of interest. Our work is motivated by the observation that, there's a cap on the problem size that the current VQMC method is capable of dealing with; to the best of our knowledge, it is applied to problems only up to hundreds of qubits. However, in many real applications, such as quantum chemistry, we may need to deal with a many-body Schrodinger equation describing interactions within molecules consisting of a very large number of orbiting electrons. The main objective of our work is to discover the potential of VQMC algorithm, and develop technologies to extend its application to a wider range of problems. We start with the classical combinatorial optimization problems such as MaxCut, and develop evolution

strategies to improve the state-of-the-art performance for larger-scale problems. We then consider a special class of problems by assuming the presence of a sparse Hamiltonian ensembles that admit a certain task regularity, and show that one can accelerate the training of VQMC and improve the convergence on new learning tasks, by employing information from previously encountered tasks through meta-learning. After that, we rework on the architecture of the model by switching from RBM with Markov chain Monte Carlo (MCMC) sampling to the one that directly supports auto-regressive sampling, achieving substantially better scalability up to problems of ten-thousand dimensions. Finally, we successfully apply our developed algorithm to quantum chemistry and financial derivative pricing. As a side project, we develop multi-fidelity active learning as a efficient sampling method to potentially improve the training of VQMC in higher dimensions.

The following is a brief overview of the chapters in this thesis.

Chapter 2: Natural Evolution Strategy. We introduce a notion of quantum natural evolution strategies, which provides a geometric synthesis of a number of known quantum/classical algorithms for performing classical black-box optimization. The algorithmic framework is illustrated for approximate combinatorial optimization problems, and a systematic strategy is found for improving the approximation ratios. In particular, it is found that natural evolution strategies can achieve approximation ratios competitive with widely used heuristic algorithms for Max-Cut, at the expense of increased computation time.

Chapter 3: Meta Learning. Motivated by close analogies between meta reinforcement learning (Meta-RL) and variational quantum Monte Carlo with disorder, we propose a learning problem and an associated notion of generalization, with applications in ground state determination for quantum systems described by random Hamiltonians. Specifically, we interpret the Hamiltonian disorder as task uncertainty for a Meta-RL agent. A model-agnostic meta-learning approach is proposed to solve the associated learning problem and numerical experiments in disordered quantum spin systems indicate that the resulting Meta Variational Monte Carlo accelerates training and improves convergence.

Chapter 4: Autoregressive Flows. While VQMC has been applied to solve high-dimensional problems, it is known to be difficult to parallelize, primarily owing to the Markov chain Monte Carlo (MCMC) sampling step. More specifically, VQMC targets the ground eigenstate by performing alternating steps of Monte Carlo sampling from a high-dimensional quantum state followed by gradient-based optimization. MCMC sampling limits the scalability of VQMC in two ways: (1) the burn-in process is an inherently sequential task; (2) sampling precise and uncorrelated samples become increasingly difficult for large input dimension. We explore the scalability of VQMC when autoregressive models are used

in place of MCMC. Autoregressive models, in contrast, provide efficient and exact computations for both sampling and density evaluation. We undertake a parallelization study of autoregressive neural quantum states, thereby improving the time-efficiency and scalability of VQMC. In particular, we demonstrate that our method scales up to ten-thousand dimensional combinatorial optimization problems.

Chapter 5: Application: Quantum Chemistry. Significant scalability challenges arise when applying variational optimization with neural-network representation on quantum states to solve interacting fermionic problems of larger scales, which correspond to non-locally interacting electronic Hamiltonian as a sum of more than thousands of products of Pauli operators. We introduce scalable parallelization strategies to improve neural-network-based VQMC calculations for ab-initio quantum chemistry applications. We establish GPU-supported local energy parallelism to compute the optimization objective for hamiltonians of potentially complex molecules. Using autoregressive sampling techniques, we demonstrate systematic improvement in wall-clock timings required to achieve CCSD baseline target energies. The performance is further enhanced by accommodating the architecture of molecular hamiltonian with the autoregressive sampling ordering. Our algorithm achieves remarkable performance in comparison with the classical approximate methods and exhibits both running time and scalability advantages over existing neural-network based methods.

Chapter 6: Application: Financial Derivative Pricing. Variational quantum Monte Carlo (VQMC) combined with neural-network quantum states offers a novel angle of attack on the curse-of-dimensionality encountered in a particular class of partial differential/difference equations (PDEs). We present a simple generalization of VQMC applicable to arbitrary linear PDEs, showcasing the technique in the Black-Scholes equation for pricing European contingent claims dependent on many underlying assets. More precisely, we adapt the McLachlan variational principle to handle non-unitary evolution according to the general linear partial differential equation, and represent the state of the system relative to a predefined basis using the output of a generative neural network, according to which exact sampling can be performed efficiently. We also propose additional modifications to the McLachlan to accomodate boundary conditions. Our proposed VQMC solver enjoys the exponential speedup for state evolution and moreover does not suffer from the exponential overhead of the readout step, since the VQMC model permits efficient queries to arbitrary probability amplitudes.

Chapter 7: Multi-Fidelity Active Learning. The performance of VQMC is substantially influenced by the batch size of the samples used to approximate the optimization objective. Both MCMC and autoregressive sampling sample a batch of samples strictly according to

the probability inferred by the model at every iteration. In addition, our finance application requires the model to pre-train on the initial condition of the PDE, involving standard training on the batch of randomly selected data points within the domain, which could be problematic in higher dimensions. Therefore, we need an data sampling mechanism that optimizes the performance of the training. Active learning (AL) is a well-established machine learning method that iteratively selects the most informative sample points based on the current model estimation. AL is particularly effective in learning from high-dimensional problems where the data points are sparse, making learning the interpolation area increasingly difficult. We make a step further by developing a general framework for training the surrogate classifier with multiple information sources, where Multi-Fidelity Active Learning (MFAL) algorithm is proposed to efficiently sample training data that maximizes the surrogate's performance under a limited query budget. Unlike the existing approaches, the proposed methodology makes no assumptions on the information sources and applies to problems with high-dimensional input and large training sets. We analyze our algorithm on examples ranging from synthetic to physics-based simulation models. and show it outperforms several other baselines by a noticeable margin in terms of accuracy and efficiency.

Chapter 2

Variational Quantum Monte Carlo and Natural Evolution Strategies

An evolution strategy [115, 106] is a black-box optimization algorithm that iteratively updates a population of candidates within the feasible region of the search space. The population is updated by a process of random mutation, followed by fitness evaluation, and subsequent recombination of best-performing members to form the next generation. The focus of this section is on the natural evolution strategies (NES) algorithm [140] and its quantum variants, in which the population is represented by a smoothly parameterized family of search distributions defined on the search space. The mutation is achieved by sampling new candidates from the search distribution, which yields a gradient estimator of the expected fitness. The recombination step involves updating the parameters of the search distribution in the direction of the steepest ascent, with respect to the information geometry implicit in the choice of search distribution.

Natural evolution strategies have recently demonstrated considerable progress in solving black-box optimization problems in high dimensions, including continuous optimization problems relevant to reinforcement learning [112]. Comparatively, little work has been done on the discrete optimization side (see however [80, 81, 96]). It was very recently shown by [47] that techniques from quantum variational Monte Carlo (VQMC) literature [24] can be adapted for approximate heuristic solution of combinatorial optimization problems. Meanwhile, in the quantum computation literature, considerable effort has focused on approximate heuristic solution of combinatorial optimization problems using low-depth quantum circuits [36, 145].

The unifying principle shared by the above algorithms is their utilization of Monte-Carlo samples drawn from a particular probability distribution, the choice of which is determined by a local optimization problem over a variational family of search distributions. The algorithms differ in the choice of variational family, as well as the geometry underlying the local optimization problem, and the use of quantum or classical resources to perform

sampling. We provide a unified view of the relationship between the geometries which underpin these algorithms. In addition, we revisit the experiments of [47] in which heuristic algorithms were found to outperform NES both in terms of approximation ratio and time to solution. In contrast, we show that NES can achieve approximation ratios for Max-Cut competitive with widely used algorithms, albeit at the expense of significantly increased computational cost. The key factor impacting performance is identified to be the batch size of the stochastic gradient estimator.

The chapter is structured as follows. First, the basics ingredients of both VQMC and the natural evolution algorithm are reviewed. Next, we clarify the relationship between natural evolution strategies and quantum approximate optimization including the classical/quantum hybrid approach introduced in [47]. Finally, numerical experiments are presented, focusing on the problem of combinatorial optimization for the Max-Cut problem.

2.1 Quantum Basics

The study of physics depends on the problem domain. In classical physics, we study "large" particles in everyday life, and the properties of the particle are continuous. In quantum physics, the objects are very small particles and the property becomes quantized. For example, an electron presents in discrete energy states, *e.g.*, ground state and first excited state. One well-known challenge in quantum many-body problems is the underlying exponentially large dimensionality. In particular, the large dimensionality lies in both the description of the problem, *i.e.* the Hamiltonian, and the solutions, *i.e.*, the eigenstates of the Hamiltonian. A quantum state ψ is in a Hilbert space

$$|\psi\rangle \in \mathcal{H} \tag{2.1}$$

admitting a linear vector form. For example, if a two-dimensional $\mathcal{H}$ is spanned by $|0\rangle$ and $\langle 1|$, and $|v\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle$, then we write ket as a column vector and bra as a row vector,

$$|v\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \\ \frac{i}{\sqrt{2}} \end{pmatrix}, \quad \langle v| = \left(\frac{1}{\sqrt{2}}, \frac{-i}{\sqrt{2}} \right). \tag{2.2}$$

To describe the quantum state, we often use the Hamiltonian description where we have a PDE or some equation describing the state, $H|\psi\rangle = \lambda|\psi\rangle$, where λ is an eigenvalue. The quantized states correspond to the eigenvalues, for example, the ground state relates to the lowest eigenvalue.

The physical observables of a given quantum system—such as the spin of an electron—are encoded as Hermitian operators on the Hilbert space describing the quantum states of the system. The actual values that this observable may take are encoded in the eigenvalues of the operator; for example, an observable describing the spin of an electron would have eigenvalues 1 and -1, which represent the spin-up and spin-down states, respectively. From this point, choosing two eigenvectors to associate with the eigenvalues determines the resulting Hermitian.

We may choose $|0\rangle$ to be the eigenvector associated with 1 and $|1\rangle$ to be the eigenvector associated with -1, in which case the Hermitian operator will have the following matrix in the $\{|0\rangle, |1\rangle\}$-basis:

$$Z = \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Choosing $|+\rangle$ and $|-\rangle$ as eigenvectors gives the operator

$$X = \sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

and choosing the states corresponding with the positive and negative y-axis on the Bloch sphere gives the operator

$$Y = \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}.$$

Physically, these choices of eigenvectors correspond with the measurement of an electron's spin along different spatial coordinate axes, as done in the Stern–Gerlach experiment. These matrices σ_x, σ_y, and σ_z are called the *Pauli spin matrices*, which are of fundamental importance in quantum mechanics as they, along with the identity matrix, form a basis for the (real) vector space of 2×2 Hermitian matrices. Therefore every observable on the state space of a single qubit may be uniquely expressed as a real linear combination of the identity matrix and the Pauli matrices.

2.2 Variational Quantum Monte Carlo

2.2.1 Formulation

We are interested in heuristic approximation algorithms for determining a minimal eigenpair of certain large and sparse random conjugate-symmetric (Hermitian) matrices that admit

an efficient description. For simplicity, throughout this chapter, we restrict to the case of real symmetric matrices for simplicity and consider only real eigenvectors. In addition, the ground eigenvector is non-negative entry-wise if all off-diagonal entries of H are further restricted to be non-positive, as a consequence of the Perron-Frobenius theorem. The sparsity assumption underlying matrices under consideration is summarized by the following requirement

Definition 2.2.1. A symmetric matrix $H \in \mathbb{R}^{N\times N}$ is row s-sparse if the list of non-zero entries $\{(y, H_{xy}) : H_{xy} \neq 0\}$ is computable in time $O(s)$, for each row index $x \in [N]$.

We consider s-sparse quantum many-body Hamiltonians where $s = O\big(\operatorname{poly}(\log(N))\big)$, although the techniques we discuss do not require an exponential separation between the sparsity parameter s and the matrix side length N. The size of these matrices is a power of 2, that is, $N = 2^n$, and they have sparsity parameter $s = \operatorname{poly}(n)$ with $n = O(\log N)$. These include as a special case quadratic unconstrained binary optimization (QUBO) problems such as Max-Cut [18]. We describe the differentiable family of trial vectors $\theta \in \mathbb{R}^d$ with a function $\psi_\theta : [N] \to \mathbb{R}$, of which the outputs are the components of the vector relative to the standard basis $\psi_\theta(x) = \langle e_x, \psi_\theta\rangle$. Given a row-sparse symmetric matrix H, we define the variational Monte Carlo learning problem as the following continuous stochastic optimization task,

$$\min_{\theta\in\mathbb{R}^d} L(\theta), \quad L(\theta) = \frac{\langle\psi_\theta, H\psi_\theta\rangle}{\langle\psi_\theta,\psi_\theta\rangle} = \mathop{\mathbb{E}}_{x\sim\pi_\theta}\left[\frac{(H\psi_\theta)(x)}{\psi_\theta(x)}\right] \geq \lambda_{\min}(H), \tag{2.3}$$

where the population quantity is computed over the probability distribution

$$\pi_\theta(x) = \frac{\psi_\theta(x)^2}{\langle\psi_\theta,\psi_\theta\rangle}. \tag{2.4}$$

It is convenient to express the population objective function as the mean of a random variable as $L(\theta) = \mathbb{E}_{x\sim\pi_\theta}[l_\theta(x)]$ where we have defined the following stochastic objective function, which is defined for $x \in [N]$ whenever $\psi_\theta(x) \neq 0$,

$$l_\theta(x) = \frac{(H\psi_\theta)(x)}{\psi_\theta(x)}, \tag{2.5}$$

and whose variance under π_θ is given by,

$$\operatorname*{var}_{x\sim\pi_\theta}\big(l_\theta(x)\big) = \mathop{\mathbb{E}}_{x\sim\pi_\theta}\big[\big(l_\theta(x) - L(\theta)\big)^2\big] = \frac{\langle\psi_\theta, H^2\psi_\theta\rangle}{\langle\psi_\theta,\psi_\theta\rangle} - \left[\frac{\langle\psi_\theta, H\psi_\theta\rangle}{\langle\psi_\theta,\psi_\theta\rangle}\right]^2. \tag{2.6}$$

It follows from the Rayleigh-Ritz principle that if the trial vector ψ_θ approaches any eigenvector of H, then the variance of stochastic objective approaches zero. In practice, the objective function is optimized using a variant of stochastic mini-batch gradient descent closely related to the stochastic natural gradient called stochastic reconfiguration [124]. Stochastic estimators for the gradient and the Fisher information matrix follow from their population forms,

$$\begin{aligned} \nabla L(\theta) &= 2 \mathop{\mathbb{E}}_{x\sim\pi_\theta}\big[\big(l_\theta(x) - L(\theta)\big)\nabla_\theta \log|\psi_\theta(x)|\big], \\ I(\theta) &= \mathop{\mathbb{E}}_{x\sim\pi_\theta}\big[\nabla_\theta \log \pi_\theta(x) \otimes \nabla_\theta \log \pi_\theta(x)\big], \end{aligned} \tag{2.7}$$

where we used an identity for the derivative of the logarithm. If the normalizing constant $\langle\psi_\theta, \psi_\theta\rangle$ of the probability distribution π_θ is unknown, then above expectation values can be approximated by the Markov chain Monte Carlo method. If the normalization condition $\langle\psi_\theta, \psi_\theta\rangle = 1$ is fulfilled, then by absorbing the factor of 2 into the logarithm one finds the reinforce gradient with baseline given by $L(\theta)$,

$$\nabla L(\theta) = \mathop{\mathbb{E}}_{x\sim\pi_\theta}\big[\big(l_\theta(x) - L(\theta)\big)\nabla_\theta \log \pi_\theta(x)\big]. \tag{2.8}$$

Markov chain Monte Carlo Sampling (MCMC) methods have been developed for sampling from a probability distribution π_θ that is difficult to directly draw *i.i.d.* samples from. The canonical Metropolis-Hastings algorithm [59] and its numerous variations, *e.g.*, Gibbs sampling [42], Reversible Jump MCMC [49] and Hamiltonian Monte Carlo [33, 61], achieve this by carefully constructing a transition kernel $p(x_{t+1}|x_t)$ for an ergodic Markov chain whose state distribution limits to the target distribution. Using samples from this Markov chain, we can then compute estimates for the expected values required in VQMC framework

$$\mathop{\mathbb{E}}_{x\sim\pi_\theta}[\phi(x)] \approx \overline{\phi}_T = \frac{1}{T}\sum_{t=1}^{T}\phi(x_t), \tag{2.9}$$

where ϕ represents some deterministic function. Furthermore, these estimates are guaranteed to be asymptotically unbiased by the ergodic theorem.

2.2.2 Model

Restricted Boltzmann Machine (RBM) is a natural choice of modeling the wavefunction ψ. RBM is a model N visible input nodes $\sigma = (\sigma_1, ..., \sigma_N)$ and M hidden nodes $\boldsymbol{h} =$

$(h_1, ..., h_M)$, where $h_i, \sigma_j \in \{-1, 1\}$ are binary variables. An energy function is defined for a configuration of nodes

$$E(\sigma, \boldsymbol{h}) = -\sum_j a_j \sigma_j - \sum_i b_i h_i - \sum_{ij} h_i W_{ij} \sigma_j = -\sigma^T a - b^T \boldsymbol{h} - \sigma^T W^T \boldsymbol{h}. \quad (2.10)$$

The model for each σ can be simplified to (by adding up all 2^M of h_i)

$$\begin{aligned} \psi(\sigma) &= \sum_{\boldsymbol{h}} e^{-E(\sigma, \boldsymbol{h})} = \sum_{\boldsymbol{h}} e^{\sigma^T a + b^T \boldsymbol{h} + \sigma^T W^T \boldsymbol{h}} \\ &= e^{\sigma^T a} \sum_{\boldsymbol{h}} e^{(b^T + \sigma^T W^T)\boldsymbol{h}}. \end{aligned} \quad (2.11)$$

Define $c = b + W\sigma$, then we have

$$\begin{aligned} \psi(\sigma) = e^{\sigma^T a} \sum_{\boldsymbol{h}} e^{\sum_i c_i h_i} = e^{\sigma^T a} \prod_i (e^{c_i} + e^{-c_i}) = e^{\sigma^T a} \prod_i 2\cosh(c_i) \\ \log \psi(\sigma) = \sigma^T a + \sum_i \log(2\cosh(c_i)). \end{aligned} \quad (2.12)$$

We do not implement the logarithm directly for table implementation, instead, we consider

$$\log(2\cosh(x)) = \log \frac{1 + e^{-2x}}{e^{-x}} = \log(1 + e^{-2x}) - (-x) = \textit{softplus}(-2x) + x. \quad (2.13)$$

The forward pass of the model is summarized below. Note that we centralize the output by the $\log(2)$ shift.

Algorithm 1 RBM

Input: Configuration σ of size N.
Output: Logarithm of the unnormalized probability.
Model Parameters: Linear operator $\mathcal{C}$ with bias. Linear operator $\mathcal{A}$ with no bias.

1: Compute $c = \mathcal{C}(\sigma)$.
2: Compute $\textit{lncosh}(c_i) = \textit{softplus}(-2c_i) + c_i - \log(2)$, for $i \in \{1, ..., N\}$
3: $\log \psi = \mathcal{A}(\sigma) + \sum_i \textit{lncosh}(c_i)$.

The classical take of RBM is used to learn the probability distribution, which means the function value is real and always equal or larger than zero. However, we want the model to be capable of representing quantum states with complex values (*i.e.*the eigenvectors of certain ground state contains negative or even complex entry values). To this end, we build two models with real parameters, outputting the real and complex values respectively. More

specifically, we have parameters $W^{real}, b^{real}, a^{real}, W^{imag}, b^{imag}, a^{imag}$. And we have

$$\begin{aligned} c^{real} &= W^{real}\sigma + b^{real},\ c^{imag} = W^{imag}\sigma + b^{imag}, \\ \log\psi(\sigma) &= \sigma^T a^{real} + i\sigma^T a^{imag} + \sum_i \log(2\cosh(c_i^{real} + ic_i^{imag})). \end{aligned} \tag{2.14}$$

The following example illustrates how the complex restricted Boltzmann machine can efficiently represent probability distributions that are hard to represent using classical neural networks. Let $X = \{0,1\}^n$ be the set of all bitstrings of length n and denote $P_n \subseteq X$ the subset consisting of all 2^{n-1} bitstring with even Hamming weight. Define $p_n \in \mathcal{P}(X)$ as follows,

$$p_n = \frac{1}{|P_n|}\mathbb{1}_{P_n}, \tag{2.15}$$

where $\mathbb{1}_{P_n} : X \to \{0,1\}$ is the indicator function for the subset P_n. Approximate representation of p_n by a real-valued restricted Boltzmann machine requires a number m of hidden units scaling exponentially in n [93] (see also [126]). In contrast, p_n can be exactly represented by a complex RBM with $m = 1$ hidden unit.

2.3 Natural Gradient Descent

Stochastic gradient descent (SGD) is the simplest but most influential algorithm for the nonlinear optimization problem. However, SGD for learning quantum amplitude is inefficient as it does not take into account the geometry of the parameter space, whereas a small change in some model parameters may have a drastically different output. One should take this into account while moving in the parameter space to find optimal parameters.

Consider the problem of optimizing a real-valued, but otherwise arbitrary function $f \in \mathbb{R}^X = \{X \to \mathbb{R}\}$ defined on a search space X. For simplicity of presentation, we focus on $|X| < \infty$, although the method generalizes in a straightforward way to infinite search spaces, as required for continuous optimization. Now consider the probability simplex $\mathcal{P}(X)$ defined as,

$$\mathcal{P}(X) = \left\{p \in \mathbb{R}^X : p \succeq 0, \sum_{x\in X} p(x) = 1\right\}. \tag{2.16}$$

The natural evolution strategies algorithm can be motivated by the following two observations, which concern the convex and Riemannian geometry of the set $\mathcal{P}(X)$:

1. The optimization problem admits the following equivalent convex relaxation,

$$\min_{x\in X} f(x) = \min_{p\in\mathcal{P}(X)} \left(\mathop{\mathbb{E}}_{x\sim p}[f(x)]\right) , \tag{2.17}$$

whose global minimizers form the subsimplex consisting of the following convex hull of Dirac distributions,

$$\operatorname{conv}\left\{\delta_a \in \mathcal{P}(X) : a \in \arg\min_{x\in X} f(x)\right\}. \tag{2.18}$$

2. There is a natural Riemannian metric called the Fisher-Rao metric defined on the interior $\operatorname{int}(\mathcal{P}(X))$ of the probability simplex, consisting of strictly positive probability vectors. The distance between $p, q \succ 0$ is defined as

$$d_{\mathrm{FR}}(p,q) := \arccos\left(\langle\sqrt{p}, \sqrt{q}\rangle\right), \tag{2.19}$$

where $\sqrt{p}$ denotes the elementwise square root of the probability vector p.

Given a smoothly parametrized family of search distributions $\{p_\theta : \theta \in \mathbb{R}^d\} \subseteq \mathcal{P}(X)$, one obtains a trivial variational bound as follows,

$$\min_{x\in X} f(x) \leq \min_{\theta\in\mathbb{R}^d}\left(\mathop{\mathbb{E}}_{x\sim p_\theta}[f(x)]\right) =: \min_{\theta\in\mathbb{R}^d} L(\theta). \tag{2.20}$$

Natural evolution strategies seek to obtain a good approximation ratio by optimizing the above variational upper bound using Riemannian gradient descent in the geometry induced by the Fisher-Rao metric, otherwise known as natural gradient descent [5]. Specifically, given a learning rate $\eta > 0$ and initial condition $\theta_0 \in \mathbb{R}^d$, one considers the deterministic sequence in $\mathbb{R}^d$ defined by,

$$\theta_{t+1} = \arg\min_{\theta\in\mathbb{R}^d}\left[\langle\theta - \theta_t, \nabla L(\theta_t)\rangle + \frac{1}{2\eta}\|\theta - \theta_t\|^2_{I_{\theta_t}}\right], \tag{2.21}$$

where I_θ denotes the Fisher information matrix, evaluated at the parameter vector $\theta \in \mathbb{R}^d$,

$$I_\theta = \mathop{\mathbb{E}}_{x\sim p_\theta}\left[\nabla_\theta \log p_\theta(x) \otimes \nabla_\theta \log p_\theta(x)\right]. \tag{2.22}$$

Indeed, it can be shown that I_θ is the local coordinate representation of the Riemannian metric tensor induced by (2.19), and the restriction to the interior of the probability simplex ensures that the logarithm is defined[1].

Observe that the iteration (2.21) defining the sequence $(\theta_t)_{t\geq 0}$ involves the unknown function f. The natural evolution strategies can now be defined as the randomized algorithm inspired by (2.21), in which $\nabla L(\theta)$ is replaced by a stochastic gradient estimator obtained by sampling from the search distribution p_θ. Likewise, if the Fisher information (2.22)

[1]This discussion ignored the fact that, unlike a bone fide Riemannian metric, the Fisher information matrix can be degenerate, in which case we choose the minimizer of (2.21) to be $\theta_{t+1} = \theta_t - \eta I^+_{\theta_t}\nabla L(\theta_t)$, where I^+_θ denotes the pseudo-inverse of I_θ.

cannot be evaluated in closed form, then it can be replaced by an associated estimator.

2.4 Quantum Approximate Optimization

In this section, we pedagogically review the proposal of [47] showing it to be a variant of Natural Evolution Strategies in which the optimization dynamics is modified as a consequence of the quantum state geometry. In addition, we identify the regime in which both methods coincide. The quantization of natural evolution strategies proceeds by replacing the search space X with a complex Euclidean space,

$$\mathbb{C}^X = \mathrm{span}\{|x\rangle : x \in X\} \ , \tag{2.23}$$

whose orthonormal basis elements are $|x\rangle$. Moreover, the probability simplex $\mathcal{P}(X)$ is replaced by the convex set of density operators,

$$\mathcal{D}(\mathbb{C}^X) = \{\rho \in \mathrm{Herm}(\mathbb{C}^X) : \rho \succeq 0,\ \mathrm{tr}(\rho) = 1\} \ , \tag{2.24}$$

where $\mathrm{Herm}(\mathbb{C}^X)$ denotes the set of Hermitian operators on $\mathbb{C}^X$. It is clear that any classical probability distribution $p \in \mathcal{P}(X)$ can be encoded as the following diagonal density operator $\mathrm{diag}(p) := \sum_{x \in X} p(x)|x\rangle\langle x|$. It is equally clear that this does not extinguish the space of admissible density operators: another possibility being the rank-1 projection operator $P_\psi = |\psi\rangle\langle\psi|/\langle\psi|\psi\rangle$ onto the one-dimensional subspace spanned by the vector $\psi \in \mathbb{C}^X$. Any admissible density operator $\rho \in \mathcal{D}(\mathbb{C}^X)$ gives rise to a valid probability distribution, which we call $\mathrm{diag}(\rho) \in \mathcal{P}(X)$, obtained from the diagonal matrix representation in the standard basis.

It is conceptually useful to introduce the following Hermitian operator $H_f \in \mathrm{Herm}(\mathbb{C}^X)$ (diagonalized by the standard basis),

$$H_f(x) := \sum_{x \in X} f(x)|x\rangle\langle x| \ , \tag{2.25}$$

whose ground-state subspace encodes the solution of the optimization problem,

$$\mathrm{span}\left\{|a\rangle : a \in \arg\min_{x \in X} f(x)\right\} \ . \tag{2.26}$$

Then for any density operator $\rho \in \mathcal{D}(\mathbb{C}^X)$ we see that the quantum expectation value of H_f

evaluated in the state ρ, is computed by the following classical expectation value,

$$\mathrm{tr}(\rho H_f) = \mathop{\mathbb{E}}_{x\sim \mathrm{diag}(\rho)} [f(x)] \quad , \tag{2.27}$$

which is an obvious upper bound for $\min_{x\in X} f(x)$. The above identity demonstrates the widely known fact that for diagonal operators of the form (2.25), unconstrained optimization over the space of quantum states $\mathcal{D}(\mathbb{C}^X)$ is equivalent to optimization over the probability simplex $\mathcal{P}(X)$. Gomes et al. [47] asks if constrained optimization within a parametrized subset of density operators provides a useful heuristic for approximate combinatorial optimization. In particular, they consider the case of rank-1 projectors, for which there exists a natural Riemannian metric called the Fubini-Study metric defined as follows,

$$d_{\mathrm{FS}}(P_\psi, P_\phi) := \arccos\left(\sqrt{\mathrm{tr}(P_\psi P_\phi)}\right) \quad . \tag{2.28}$$

Thus, given a smoothly parametrized subset $\{\psi_\theta : \theta \in \mathbb{R}^d\} \subseteq \mathbb{C}^X$ of a complex Euclidean space, one can define quantum natural evolution strategies as the local optimization of the following variational upper bound, via Riemannian gradient descent in the geometry induced by the Fubini-Study metric,

$$\min_{x\in X} f(x) \leq \min_{\theta\in\mathbb{R}^d} \left(\mathop{\mathbb{E}}_{x\sim|\psi_\theta|^2} \big[f(x)\big] \right) \quad . \tag{2.29}$$

The choice to restrict to rank-1 projection operators involves no loss of generality compared to classical natural evolution strategies because if we choose $\psi_\theta = \sum_{x\in X} \sqrt{p_\theta(x)}|x\rangle$, then the Fubini-Study geometry reduces to Fisher-Rao. Thus, if the parametric family is chosen to be strictly positive $\psi_\theta \succ 0$, then Riemannian gradient descent in the Fubini-Study geometry coincides with natural gradient descent[2] and we recover classical natural evolution strategies. Therefore we henceforth use the terminology 'natural gradient' to refer to both geometries interchangeably.

The natural gradient has been thoroughly explored in the variational Monte Carlo for ground-state optimization of non-diagonal Hermitian operators [124, 24], and more recently in the variational quantum algorithm literature [147, 127, 71]. Finally, we note the possibility of generalizing to higher-rank density operators, the investigation of which is left to future work.

[2]The local coordinate representation of the Fubini-Study and Fisher-Rao metric tensor agree if the Berry connection vanishes [127].

2.5 Experiments

Consider combinatorial optimization problems defined on $X = \{\pm 1\}^n$. For concreteness we focus on the Max-Cut problem corresponding to an undirected graph $G = (V, E)$ of size $|V| = n$. The function $f \in \mathbb{R}^X$ to be minimized is simply,

$$f(x) = \sum_{\{i,j\} \in E} \frac{x_i x_j - 1}{2} \ , \tag{2.30}$$

where $-f(x)$ is the size of the cut corresponding to the configuration $x \in X$. After fixing a parametrized family of wavefunctions, we locally optimize the variational upper bound (2.29) using stochastic natural gradient descent. Following [47], we choose the variational wavefunction $\psi_\theta \in \mathbb{C}^X$ to be of Boltzmann form [24],

$$\psi_\theta(x) = \sum_{z \in \{\pm 1\}^m} \exp\left[\langle z, Wx + b\rangle + \langle x, c\rangle\right] \ , \tag{2.31}$$

where the variational parameters $\theta = (W, b, c) \in \mathbb{F}^{m \times n} \times \mathbb{F}^m \times \mathbb{F}^n$ and $\mathbb{F}$ denotes either $\mathbb{R}$ or $\mathbb{C}$. In the case $\mathbb{F} = \mathbb{R}$ we have $\psi_\theta \succ 0$ and the optimization problem is equivalent to natural evolution strategies [140]. An example illustrating the increased expressiveness of complex restricted Boltzmann machines compared to their real-valued counterparts for representing classical probability distributions is also presented.

Algorithm Performance Comparisons

# Nodes / # Cut	50	70	100	150	200	250
Cut Number						
Random	149.60 ± 7.41	297.10 ± 11.48	614.30 ± 16.94	1436.80 ± 27.14	2467.30 ± 30.27	3888.00 ± 39.06
GW	203.40 ± 3.61	380.90 ± 8.48	752.50 ± 9.22	1685.70 ± 13.10	2875.10 ± 22.34	4439.90 ± 26.07
BM	206.30 ± 0.46	390.90 ± 0.54	776.60 ± 1.56	1719.90 ± 1.58	**2931.20 ± 8.07**	**4526.70 ± 12.96**
NES	**206.97 ± 0.01**	**392.98 ± 0.01**	**777.62 ± 1.40**	**1721.84 ± 7.21**	2927.82 ± 12.10	4515.72 ± 11.41
Time elapsed (sec)						
Random	*	*	*	*	*	*
GW	0.32 ± 0.11	0.51 ± 0.04	1.59 ± 0.02	5.34 ± 0.20	12.24 ± 0.34	21.09 ± 0.54
BM	0.77 ± 0.12	0.96 ± 0.08	1.36 ± 0.08	1.52 ± 0.16	2.32 ± 0.13	2.63 ± 0.14
NES	964.20 ± 12.58	1883.46 ± 20.39	3683.93 ± 65.30	8606.25 ± 203.14	15693.06 ± 431.68	23621.38 ± 847.84

Table 2.1: Performance comparison for algorithms on graph instances of different sizes. The mean and std are computed from 10 trials with different random seeds. Here, "*" indicates that the time elapsed is less than 0.1 seconds.

Although no polynomial-time algorithm is known for solving Max-Cut on general graphs, many approximation algorithms have been developed in the past decades. Random Cut Algorithm is a simple randomized 0.5-approximation algorithm that randomly assigns

each node to a partition (e.g., see [92]). [46] improved the performance ratio from 0.5 to at least 0.87856, by making use of the semidefinite programming (SDP) relaxation of the original integer quadratic program. [22] reformulated the SDP for Max-Cut into a non-convex problem, with the benefit of having a lower dimension and no conic constraint, but with the disadvantage of being heuristic in nature.

The above heuristic and approximate algorithms were used as benchmark solvers for comparison with quantum natural evolution strategies. The implementation of Goemans-Williamson Algorithm used the `CVXPY` [30, 3] package and the Burer-Monteiro reformulation with the Riemannian Trust-Region method [2] used `Manopt` toolbox [16], which essentially implements the optimization algorithm proposed by [65].

The classical and quantum variants of natural evolution strategies were realized using the variational Monte Carlo (VMC) [88] method with Stochastic Reconfiguration (SR) [124], as implemented in the `NetKet` toolbox [25]. The SR optimization was performed using a regularization parameter $\lambda = 0.1$ and a learning rate $\eta = 5 \times 10^{-2}$, for 90 iterations. At each iteration, the number of Monte Carlo sampled observables (batch size for training) is 4096. The mean performance of the observable batch drawn from the trained model is reported. For Restricted Boltzmann Machine (RBM) model, the number of hidden variables is set to be the same as the number of spins; the weights are complex-valued, initialized with Gaussian distribution of mean 0 and standard deviation (std) 0.01. Throughout the experiments, the timing benchmarks are performed on a core of an 8-core processor, Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz, with 128 GB of memory.

For evaluation, we constructed a problem instance for each graph size n by randomly generating graph Laplacians with edge density 50%, for $n \in \{50, 70, 100, 150, 200, 250\}$. This defines a set of problem instances, indexed by n, which are held fixed throughout the experiments. For the fixed problem instance, each algorithm was executed 10 times using 10 random seeds/initializations. In Table 2.1, we report the mean and std of the performance over graph instances of different sizes. Since the optimal cut for a given problem instance cannot be computed for large scale problems, we approximate it with an upper bound $\mathrm{UBD}(I)$ [17], which is the optimal value of the SDP relaxation, according to the arguments in [46]. In Figure 2.1(L), we present the ratio $\mathrm{cut}(A(I))/\,\mathrm{UBD}(I)$ for the same algorithms A and graph instances I in Table 2.1 with box plot.

The choice of batch size was found to be a crucial factor controlling the performance of the algorithm. Intuitively, it quantifies the exploration capability in the state space: the algorithm has a better chance to discover the ground state if it is allowed to explore more. The performance (as measured by approximation ratio) as well as the training time, as a function of batch size, are reported in Figure 2.1(R) on the graph instance with 150

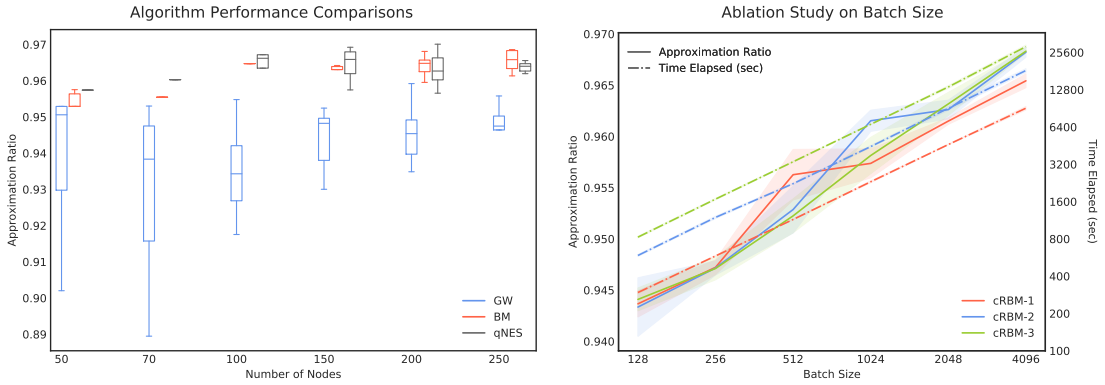


Figure 2.1: (L) The performance comparison box plot for Goemans-Williamson (GW), Burer–Monteiro (BM), and quantum Natural Evolution Strategies (qNES). The approximation ratio is approximated by dividing the upper bound from the cut number. The performance of qNES is on par with the SDP benchmarks. (R) An ablation study for the training batch size, tested on the graph instance with 150 nodes, using cRBMs with different hidden unit densities. The increase in batch size improves the performance of qNES, at a cost of the linear increase in training time.

nodes. The ablation study also reveals that increasing hidden unit density is correlated with performance, provided that training batch size is correspondingly increased. This is expected behavior since increased model capacity is required to capture the increasing complexity from the sampled observables.

Ablation Study on Optimizer and Model Architecture

Architecture \ Optimizer	Adadelta	Adamax	Momentum	RMSprop	SGD
w.o. Natural Gradient					
cRBM-1	1692.25 ± 6.13	1648.19 ± 12.67	1656.60 ± 6.83	1642.60 ± 15.07	1608.20 ± 52.05
Natural Gradient					
rRBM-1	1646.96 ± 6.80	1687.16 ± 10.34	1694.02 ± 7.71	1692.76 ± 4.79	1704.16 ± 7.89
cRBM-1	1699.74 ± 11.60	1697.04 ± 17.24	1701.32 ± 8.32	1700.39 ± 10.00	1704.06 ± 5.28
cRBM-3	1713.13 ± 8.78	1693.07 ± 5.39	1700.33 ± 6.77	1686.20 ± 10.91	1709.14 ± 8.70
FC	1700.01 ± 7.50	1697.11 ± 8.84	1702.02 ± 6.02	1702.20 ± 14.54	1702.95 ± 8.13

Table 2.2: An ablation study for different optimization algorithms and model architectures, tested on the graph instance with 150 nodes (UBD $= 1784.89$) and batch size 1024. Stochastic natural gradient descent consistently outperformed all other optimizers, across all architectures considered.

The role of optimization algorithm and model architecture was also investigated, focusing on the following optimizers (with and without natural gradient updates): Adadelta [148], Adamax ($\alpha = 5 \times 10^{-3}$) [68], Momentum ($\eta = 5 \times 10^{-2}$) [130], RMSprop ($\eta = 5 \times 10^{-3}$), SGD ($\eta = 5 \times 10^{-2}$). The architecture was chosen to be the restricted Boltzmann form (2.31) with hidden unit density $\alpha = m/n$ (RBM-α), with real-valued weights (rRBM) and complex weights (cRBM).

The natural gradient descent [5, 124] proved essential for converging to a good local

optimum; results on cRBM-1 suggest that optimizers equipped with natural gradient updates consistently outperformed those without. The use of complex RBM yielded some improvement relative to the real-valued case, although the performance gap largely disappeared when natural gradient updates were applied. In addition, we found that architectures other than RBM can achieve high performance: the single-layer perceptron (FC) is compatible with RBM across all optimizers. In general, Stochastic Natural Gradient Descent consistently achieves optimal performance over all architectures, in comparison with other optimizers.

2.6 Conclusion

The Max-Cut approximation ratio achieved by natural evolution strategies is competitive with widely used solvers, although this comes at the expense of significantly increased computation time. It will be interesting to investigate other combinatorial optimization problems, particularly from the spin-glass literature, which do not admit semidefinite program relaxations.

It is legitimate to inquire about possible advantages for classical stochastic optimization, given access to efficiently simulable subsets of quantum states, such as the complex Boltzmann machine. In the case of natural evolution strategies, our ablative study indicates that the use of natural gradient descent levels the playing field between the quantum and classical variants.

Chapter 3

Meta Variational Quantum Monte Carlo

Although deep neural networks excel in individual learning tasks, they are brittle with respect to task deformation. This fragility presents a challenge to the design of artificially intelligent agents which are required to efficiently adapt from known source tasks to a stream of unknown and dynamically changing target tasks. In order to quantify the ability of an agent to adapt when confronted with a stream of learning tasks, it proves convenient to adopt the modeling assumption in which there exists a probability distribution supported on the space of possible tasks called the task distribution. Meta-learning (also called *learning to learn* [133]) attempts to formalize the goal of adaptivity by exploiting regularities in the task distribution in order to output a hypothesis that performs well on new task realizations, given limited data access to each target task. It is instructive to contrast meta-learning with the comparatively simpler paradigm of transfer learning in which the target task is *known*. In this chapter, we explore a formal identification between model-agnostic meta-learning [37] and a seemingly unrelated problem in quantum physics involving quantum Monte Carlo with disorder.

Given a fixed target Hamiltonian H acting on a Hilbert space $\mathcal{H}$ of exponentially high dimension, variational quantum Monte Carlo (VQMC) [88] computes an estimate of its minimal eigenvalue and a description of the associated eigenvector. The ability of VQMC to overcome the curse-of-dimensionality in this context stems from a reformulation of the Rayleigh-Ritz principle as a stochastic optimization problem, the optimum of which is a function outputting the components of the ground eigenvector on some orthonormal basis. Leveraging the close connection between VQMC and deep reinforcement learning, Carleo and Troyer [24] showed that when neural networks are exploited as trial functions and optimized using natural gradient techniques, VQMC can achieve state-of-the-art results in finding the ground state energies of the antiferromagnetic Heisenberg (AFH) model, in which the Hamiltonian is defined by a geometrically local interaction graph corresponding to a two-dimensional lattice. The domain of applicability of so-called neural-network quantum states has since been expanded to encompass problems of electronic structure in

finite [95, 79, 128] and infinite dimensions [100]. Further connections between VQMC and deep learning have been elaborated in [47, 150] where it was shown that VQMC is a quantum generalization of Natural Evolution Strategies (NES) [140], which in turn, provides a single-step realization of natural policy gradient learning [66].

The ability of VQMC to obtain state-of-the-art results comes at the expense of significant computation time due to the sequential nature of the sampling process. Sharir et al. [120] proposes autoregressive modeling in replacement of the original Markov chain Monte Carlo (MCMC) to speed up the sampling, and Zhao et al. [152] focuses on the scalability of VQMC to large problems and efficient use of all available resources. This works attempts to address a similar problem but under a different context, where we assume the presence of sparse matrix ensembles that admit a certain task regularity. Our hypothesis is that one can accelerate the training of VQMC and improve the convergence of new learning tasks, by employing information from previously encountered tasks. The naive approach that uses the pre-trained model parameters from one task as the initialization on the target task fails to apply in VQMC, as there's no notion of generalization to out-of-sample data for a task of which the stochastic objective function is an unbiased estimator of a given population objective. This lack of delineation between the training and testing phase is closely analogous to deep reinforcement learning, where agents are trained and tested in the same learning environment and the algorithm typically outputs a policy that is strongly overfitted to the learning task. The above considerations motivate the viewpoint that meta-learning provides the relevant context in which to discuss generalization both for deep reinforcement learning agents and VQMC. Indeed, Meta-RL has enjoyed significant progress in the last few years, propelled by the discovery of a scalable gradient-based instantiation suitable for deep learning called model-agnostic meta-learning [37] (MAML).

In this work, we investigate the empirical performance of meta-VQMC over sparse matrix ensembles with different kinds of task regularity, which we encode via geometric locality assumptions. Our experimental results suggest that meta-VQMC is capable of effectively accelerating the training of VQMC and improving the convergence of tasks from the given ensembles. The work of this chapter expands on the preliminary work of Zhao et al. [151] in which the notion of meta-VQMC was introduced and numerically investigated for diagonal matrix ensembles as a special case, which corresponds to the Max-Cut optimization problem. The content of the chapter is organized as follows: in section 3.1, we introduce single-task variational quantum Monte Carlo, emphasizing the connection with the REINFORCE algorithm and natural policy gradient learning. The basics of meta-learning and the meta-VQMC are then recalled. The theory underlying model-agnostic meta-learning and gradient-based meta-VQMC are described in section 3.2.

Experimental results and analysis are presented in section 3.4 and section 3.5 concludes the chapter.

3.1 Background

Meta-learning. In contrast to the single-task formulation of VQMC, which accepts a fixed target Hamiltonian as input, our proposed *meta-VQMC* asks for an approximation of the ground energy for an *ensemble* of Hamiltonians H_τ indexed by a random disorder parameter τ sampled from a known distribution $\mathcal{T}$. The simplest strategy of retraining a separate neural-network quantum state from scratch for each realization of the disorder parameter τ is impractical. The goal is thus shifted to finding a neural network that is maximally adaptive to new realizations of the disorder. For experimental support, our focus in this chapter is a special case of disordered quantum spin systems. These results are viewed as a stepping stone to random electronic structures, in which we anticipate similar optimization considerations to apply.

The formulation of meta-VQMC exhibits obvious parallels with *meta-learning* or *learning-to-learn* in the machine learning literature [133], where data from previously encountered learning tasks are employed to accelerate performance on new tasks, drawn from an underlying task distribution $\mathcal{T}$. In the language of meta-learning, τ indexes the learning task, and $\mathcal{T}$ denotes the distribution over all tasks. In meta-VQMC, we assume the task distribution is known to the learner; in contrast, conventional meta-learning assumes $\mathcal{T}$ is unknown but possesses sufficient regularity to render meta-learning feasible.

Relationship with previous work. We differentiate our proposal from the uses of meta-learning that have been proposed elsewhere in the quantum information literature. In [141], for example, meta-learning has been proposed to mitigate various sources of noise, specifically shot noise and parameter noise. In the context of VQMC, shot noise is analogous to variance associated with finite mini-batches, whereas parameter noise has no clear analogue. Ref. [137] is the most similar to ours in that they consider meta-learning from known distributions. They differ by the choice to focus on variational quantum algorithms such as VQE and QAOA and by the fact that they do not use model-agnostic meta-learning. Instead, the meta-learning outer-loop involves training a separate recurrent neural network, similar to [6]. The notion of meta-VQMC and a model-agnostic training algorithm was originally introduced in [151]. The numerical experiments of [151] restricted to diagonal matrix ensembles which can be understood as classical combinatorial optimization problems. In this chapter, we expand the experiments to include off-diagonal matrix ensembles which

have no classical analog.

3.2 Theory

A simple strategy that has proven successful in meta-learning of deep neural networks is multi-task transfer learning [26, 11], which aims to learn an initialization for subsequent tasks by jointly optimizing the learning objective of multiple tasks simultaneously, using a mini-batch training strategy that interleaves batches across the tasks. Multi-task learning is, however, prone to catastrophic interference [86], making it unsuitable for generalization to the VQMC. The problem is exemplified by some of the simplest examples of disordered spin systems: suppose H_τ is a random Hamiltonian whose expected value under the disorder parameter vanishes $\mathbb{E}_{\tau\sim\mathcal{T}}[H_\tau] = 0$. As a concrete example, consider the Sherrington-Kirkpatrick Hamiltonian, in which τ represents a collection of i.i.d. centered Gaussian random variables $J_{ij} \sim N(0,1)$ representing the exchange energies. If we denote by L_τ the objective function corresponding to disorder parameter τ, then the multi-task learning objective function, expressed in the population limit, is given by

$$L_{\mathrm{MTL}}(\theta) := \mathop{\mathbb{E}}_{\tau\sim\mathcal{T}}[L_\tau(\theta)] = \mathop{\mathbb{E}}_{\tau\sim\mathcal{T}}\left\{\mathop{\mathbb{E}}_{x\sim\pi_\theta}\left[\frac{(H_\tau\psi_\theta)(x)}{\psi_\theta(x)}\right]\right\} = \frac{\langle\psi_\theta, \mathbb{E}[H_\tau]\psi_\theta\rangle}{\langle\psi_\theta,\psi_\theta\rangle} = 0 \ . \tag{3.1}$$

The fact that the multi-task learning objective loses dependence on θ in the population limit implies that the associated mini-batch algorithm makes no progress asymptotically.

In order to define an objective function that is asymptotically non-vacuous and which promotes adaptation to new realizations of disorder, we propose to optimize the following meta-learning objective function, again presented in population form for simplicity [6],

$$L_{\mathrm{ML}}(\theta) := \mathop{\mathbb{E}}_{\tau\sim\mathcal{T}}\left[L_\tau\big(U_\tau^t(\theta)\big)\right] = \mathop{\mathbb{E}}_{\tau\sim\mathcal{T}}\Big[L_\tau\big(\underbrace{U_\tau\circ\cdots\circ U_\tau}_{t\text{ times}}(\theta)\big)\Big] \ , \tag{3.2}$$

where $U_\tau^t : \mathbb{R}^d \to \mathbb{R}^d$ denotes the t-fold application of a task adaptation operator U_τ, which in the simplest case of gradient descent with step size β, is given by $U_\tau(\theta) = \theta - \beta\nabla L_\tau(\theta)$. Optimization of the meta-learning objective L_{ML} ensures that when a new realization of the disorder parameter is drawn, the initialization performs well after performing one or more steps of gradient descent. Loosely speaking, meta-learning can be justified when one has a budget for running a few steps of gradient descent.

In the case of meta-VQMC, we consider gradient-based optimization. Specifically, we focus on model-agnostic meta-learning (MAML) [37] which is a gradient-based algorithm that has been proposed for optimizing the meta-learning objective. Straightforward applica-

Algorithm 2 MAML [37] adapted to meta-VQMC (batched over tasks).

Input: Matrix ensemble $\mathcal{T}$, adaptation operator U_τ, adaptation steps t
Initialize θ
while not done **do**
  Sample batch of disorder parameters $B \overset{\text{iid}}{\sim} \mathcal{T}$
  for each disorder parameter $\tau \in B$ **do**
    $\theta_\tau = U_\tau^t(\theta)$
    $\nabla_\tau = (U_\tau^t)'(\theta) \nabla L_\tau(\theta_\tau)$
  end for
  $\nabla = \frac{1}{|B|} \sum_{\tau \in B} \nabla_\tau$
  $\theta \leftarrow \textsc{Optimizer}(\theta, \nabla)$
end while

tion of the chain rule gives rise to the following gradient estimator for the meta-learning objective,

$$\nabla L_{\text{ML}}(\theta) = \mathbb{E}_{\tau \sim \mathcal{T}} \big[(U_\tau^t)'(\theta) \nabla L_\tau \big(U_\tau^t(\theta) \big) \big] \ , \tag{3.3}$$

where $(U_\tau^t)'(\theta)$ denotes the Jacobian matrix of the function $U_\tau^t : \mathbb{R}^d \to \mathbb{R}^d$. The pseudocode for MAML is outlined in Algorithm 2. In order to facilitate readability, we have presented the algorithm with batching only in the task index, leaving the remaining expectation values (with respect to Born probabilities) in population form. In a practical algorithm, the intermediate variables θ_τ and ∇_τ are estimated stochastically using independent batches of data generated by the same task τ. Since the computation of the Jacobian involves an expensive back-propagation, first-order MAML (foMAML) has been proposed (*e.g.*, [37, 94]) as a simplification of MAML, in which the Jacobian matrix is approximated by the identity matrix.

3.2.1 An illustrative example

The fact that the meta-learning objective function manages to avoid the catastrophic interference phenomenon can be illustrated by the following toy model[1]. Rather than considering the Rayleigh quotient, consider the following ensemble of quadratic functions specified by a random positive-definite matrix $A \in \mathbb{R}^{d \times d}$ and a random vector $b \in \mathbb{R}^d$,

$$L_\tau(\theta) = \frac{1}{2} \langle \theta, A\, \theta \rangle - \langle b, \theta \rangle \ , \tag{3.4}$$

where the random variable $\tau = (A, b)$ now corresponds to the task label. In the simplest setting of single-step ($t = 1$) meta-learning with vanilla update operator $U_\tau(\theta) = \theta -$

[1]This quadratic model has also been analyzed in the context of convergence theory in [35].

$\beta\nabla L_\tau(\theta)$, the optimal solution of the multi-task and meta-learning objectives can be found in closed form,

$$\arg\min_{\theta\in\mathbb{R}^d} L_{\mathrm{ML}}(\theta) = \mathbb{E}\big[A(I-\beta A)^2\big]^{-1}\mathbb{E}\big[(I-\beta A)^2 b\big] \ . \tag{3.5}$$

In the limit $\beta \to 0$ corresponding to multi-task learning, the optimal solution is found to only depend on the mean value of the random variable τ, whereas the meta-learner (corresponding to $\beta > 0$) exploits information in the higher-order moments of τ.

3.3 Architecture

The structure of Restricted Boltzmann Machine (RBM) is

$$\begin{aligned} &\textit{Input} \xrightarrow{[bs,n]} \texttt{Fc}_{n,cn} \xrightarrow{[bs,cn]} \texttt{Lncoshsum} \xrightarrow{[bs]} \textit{Output}1 \\ &\xrightarrow{[bs,n]} \texttt{Fc}_{n,1} \xrightarrow{[bs]} \texttt{Add}\ \textit{Output}1 \xrightarrow{[bs]} \textit{Output}. \end{aligned}$$

The structure of 1D Convolutional Neural Network (CNN) is

$$\begin{aligned} &\textit{Input} \xrightarrow{[bs,n]} \texttt{Conv1d}_{c,3,1,1} \xrightarrow{[bs,c,n]} \texttt{Reshape} \\ &\xrightarrow{[bs,cn]} \texttt{Fc}_{cn,n} \xrightarrow{[bs,n]} \texttt{Lncoshsum} \xrightarrow{[bs]} \textit{Output}. \end{aligned}$$

The structure of 2D Convolutional Neural Network (CNN) is

$$\begin{aligned} &\textit{Input} \xrightarrow{[bs,\sqrt{n},\sqrt{n}]} \texttt{Conv2d}_{c,3,1,1} \xrightarrow{[bs,c,\sqrt{n},\sqrt{n}]} \texttt{Reshape} \\ &\xrightarrow{[bs,cn]} \texttt{Fc}_{cn,n} \xrightarrow{[bs,n]} \texttt{Lncoshsum} \xrightarrow{[bs]} \textit{Output}. \end{aligned}$$

Here *bs* is the batch size and n is the number of sites. $\texttt{Fc}_{a,b}$ is a fully connected layer with input size a and output size b. $\texttt{Conv}_{c,k,s,p}$ is a convolutional layer with output channel size c, kernel size k, stride s and circular padding size p. `Lncoshsum` refers to a series of linear and non-linear operations involving: 1) taking natural logarithm for each entry of the input tensor; 2) taking hyperbolic cosine for each entry of the input tensor; 3) summation over the last dimension of the input tensor.

3.4 Experiments

For our experiments, we focus on symmetric matrices whose side length is a power of 2, that is, $N = 2^n$. The real vector space of $2^n \times 2^n$ symmetric matrices contains a subspace of dimension $O(\mathrm{poly}(n))$ which is parametrized by real parameters $g_i, h_i, g_{ij} \in \mathbb{R}$ as follows,

$$H = -\sum_{1\le i\le n} \big(h_i X_i + g_i Z_i\big) - \sum_{1\le i<j\le n} g_{ij} Z_i Z_j, \tag{3.6}$$

where $X_i := I^{\otimes(i-1)} \otimes X \otimes I^{\otimes(n-i)}$ and $Z_i := I^{\otimes(i-1)} \otimes Z \otimes I^{\otimes(n-i)}$ are defined in terms of the following elementary 2×2 matrices,

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad , \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad , \qquad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad . \tag{3.7}$$

The hamiltonian H can be verified to be n-sparse as described in definition 2.2.1, and admits a binary representation with entry value written as

$$H_{xy} = -\sum_{1\le i\le n} \big(h_i \delta_{x_1 y_1} \cdots \delta_{\neg x_i y_i} \cdots \delta_{x_n y_n} + g_i(1-2x_i)\big) - \delta_{xy} \sum_{1\le i\le j\le n} g_{ij}(1-2x_i)(1-2x_j), \tag{3.8}$$

where the row and column indices x, y are in their binary forms $x = 2^{n-1}x_1 \cdots 2^0 x_n, y = 2^{n-1}y_1 \cdots 2^0 y_n$, and $\neg x_i$ denotes logical negation of $x_i \in \{0, 1\}$.

Given a description of the matrix ensemble and a differentiable family of trial vectors described by a neural network, we seek an initialization strategy which rapidly accelerates the convergence of the trial vector to the ground space of randomly drawn problem instances. The proposed strategy was compared against training random problem instances from scratch (randomly initialized neural network). We also consider random initialized models that are trained using the stochastic reconfiguration method (stochastic natural gradient descent), as an additional baseline. To showcase the robustness of meta-VQMC, we conduct experiments across various settings involving different task distributions and model architectures.

3.4.1 Task distribution generation

Matrix ensembles of exponential size were chosen by specifying the distribution for the $O(\mathrm{poly}(n))$ parameters $g_i, h_i, g_{i,j}$ appearing in (4.7). In this work, we impose $h_i \le 0$ to ensure that the ground eigenvector can be chosen to be a non-negative vector due to the Perron-Frobenius theorem. Meta-learning experiments were conducted using four

task distributions (matrix ensembles) supported on the subspace of symmetric matrices of the form (4.7), by first fixing a base matrix and then introducing Gaussian noise to the parameters. The first base matrix we consider is referred to as the Max-Cut problem and is defined by

$$H_{\text{Max-Cut}} = -\frac{1}{4} \sum_{1 \leq i < j \leq n} L_{ij} Z_i Z_j \ , \tag{3.9}$$

where $L = [L_{ij}]$ denotes the Laplacian matrix for an undirected graph $G = (V, E)$ of size $|V| = n$. The diagonal entries of this matrix correspond to the sizes of the 2^n possible cuts on the graph G. The adjacency matrix for G was chosen by forming the $n \times n$ matrix $(B + B^T)/2 - \textit{diag}(B)$ with entries B_{ij} sampled from the rounded $\text{Bernoulli}(0.5)$. By design, the diagonal entries of G are all zero.

The second base matrix is referred to as Sherrington-Kirkpatrick model, which is a matrix of the form (4.7) with $g_i, g_{ij} \sim U(-1, 1)$ and $h_i \sim U(0, 1)$ sampled once and fixed. The final experiments consider geometrically local versions of the Sherrington-Kirkpatrick model (referred to as transverse field Ising model), in which the interactions are determined by a one-dimensional ring geometry $\mathbb{Z}_L = \{0, \ldots, L-1\}$ (with addition defined modulo L) and a two-dimensional torus geometry $\mathbb{Z}_L \times \mathbb{Z}_L$. The respective base matrices are given by

$$H_{\text{TIM-1D}} = -\sum_{i \in \mathbb{Z}_L} \left(g_i Z_i Z_{i+1} + h_i X_i \right) \ , \tag{3.10}$$

$$H_{\text{TIM-2D}} = -\sum_{(i,j) \in \mathbb{Z}_L^2} \left(g^v_{i,j} Z_{i,j} Z_{i+1,j} + g^h_{i,j} Z_{i,j} Z_{i,j+1} + h_{i,j} X_{i,j} \right) . \tag{3.11}$$

Having fixed the base matrices as above, sampling from the respective matrix ensembles was achieved via the following procedure. In the case of Max-Cut, a random adjacency matrix was formed by perturbing the base adjacency matrix A with additive noise matrix δA and then rebinarizing the sum $A + \delta A$. In particular, we chose $\delta A = (N + N^T)/2$ where the entries of N consist of independent centered Gaussian noise of variance σ^2. In the case of the transverse field Ising and Sherington-Kirkpartric models, the parameters were perturbed by additive centered Gaussian noise of variance σ^2, followed by clippings of h_is to non-negative values.

3.4.2 Settings

Network architectures are chosen to be either restricted Boltzmann machine (RBM) or convolutional neural network (CNN). Carleo *et al.* (2017) [24] proposed RBM for the transverse field Ising model and the anti-ferromagnetic Heisenberg model, taking the one-

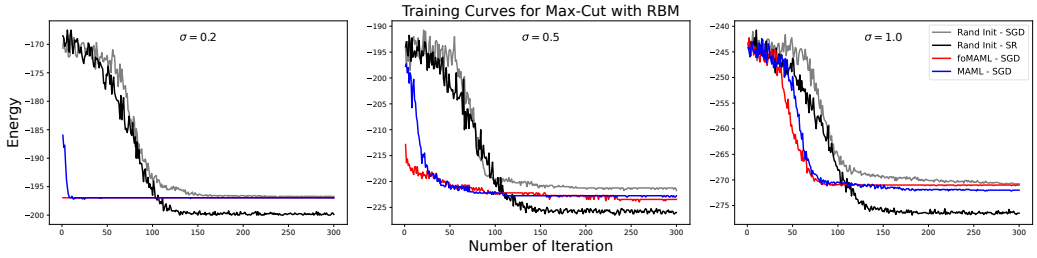


Figure 3.1: Results for RBM on $\mathcal{T}_\sigma$ with $\sigma = 0.2, 0.5, 1.0$ from left to right, where the base hamiltonian is Max-Cut-49. Each curve is the average of the training curves over 8 testing tasks randomly sampled from $\mathcal{T}_\sigma$. The learning rates for outer and inner loops are 0.002 and 0.005, respectively. Initializations from foMAML and MAML discover solutions that outperform random initialization using far fewer iterations with SGD, on problems with diagonal matrix ensembles. However, SR (stochastic reconfiguration) outperforms SGD in the long run. On the other hand, the convergence of foMAML and MAML goes slower as σ increases, indicating that task adaptation becomes more difficult for task distributions that are more complex.

dimensional state as input and outputs the logarithmic probability amplitude. Besides, RBM accommodates input states with higher dimensional structures by flattening them down to a single dimension. For Ising models with local geometric structures, it's also natural to use convolutions as local operators to process the inputs. In this chapter, we consider CNN models with one layer of convolution followed by a fully connected layer. More architectural details are deferred to the appendices.

Each iteration of the meta-learning loop involves independently sampling a batch of 16 tasks from the task distribution $\mathcal{T}_\sigma$, parametrized by σ. During testing, 8 testing tasks are sampled from $\mathcal{T}_\sigma$ and fixed for evaluation purposes. The inner loop used 1 iteration of vanilla SGD with batch size 1024, while the outer loop training used 50 iterations of vanilla SGD with batch size 16. The learning rates for outer and inner loops depend on the problem type and model architecture.

3.4.3 Analysis of results

In Figure 3.1, we train separate RBM models on task distributions $\mathcal{T}_\sigma = 0.2, 0.5, 1.0$ with initializations from MAML and foMAML methods, and plot the training curves of the models for 300 iterations. The training curves of randomly initialized models with SGD and stochastic reconfiguration are also plotted for comparison. Our result shows that models initialized using MAML discover solutions that outperform SGD in the long run using far fewer iterations than SGD, consistent with the expectation that MAML initializations perform well after performing only a few iterations of gradient descent. Despite accelerated convergence, MAML was found to converge to suboptimal local minima in this case compared to the stochastic reconfiguration (initialized from random). This observation

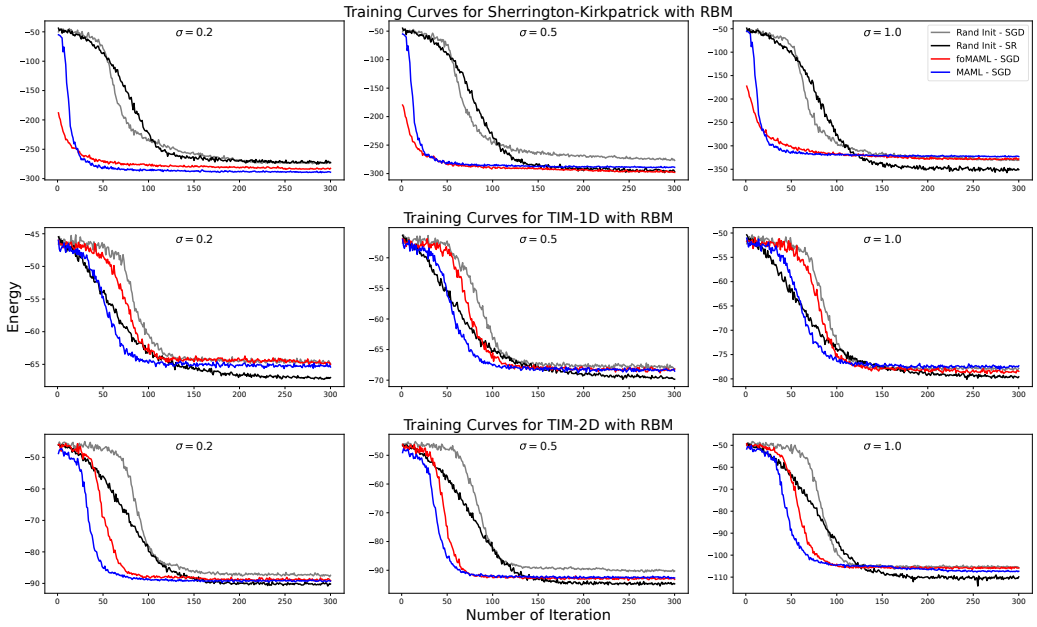


Figure 3.2: Results for RBM on $\mathcal{T}_\sigma$ with $\sigma = 0.2, 0.5, 1.0$ from left to right, where the base hamiltonians are Sherrington-Kirkpatrick model and TIM (transverse field Ising model) with 49 sites. Each curve is the average of the training curves over 8 testing tasks randomly sampled from $\mathcal{T}_\sigma$. The learning rates for outer and inner loops are 0.002 and 0.005, respectively. Initializations from foMAML and MAML discover solutions that outperform random initialization using far fewer iterations with SGD, on problems with sparse non-diagonal matrix ensembles.

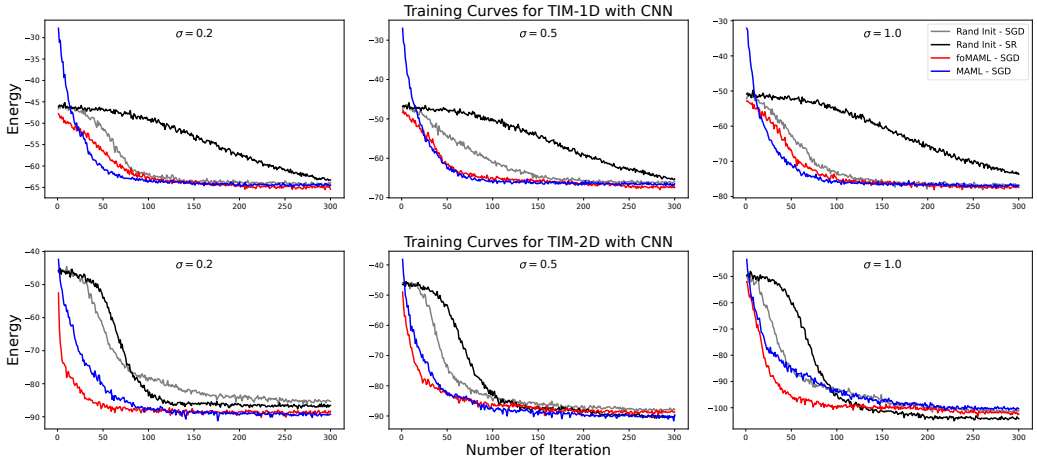


Figure 3.3: Results for CNN on $\mathcal{T}_\sigma$ with $\sigma = 0.2, 0.5, 1.0$ from left to right, where the base hamiltonians are TIM (transverse field Ising model) on 1D lattices with 49 sites and 2D lattices with 7×7 sites. Each curve is the average of the training curves over 8 testing tasks randomly sampled from $\mathcal{T}_\sigma$. The learning rates for outer and inner loops are 0.005 and 0.01, respectively. Initializations from foMAML and MAML discover solutions that outperform random initialization using far fewer iterations with SGD and CNN. On the other hand, the performance of CNN is in general comparable with RBM.

reiterates the importance of the Riemannian geometry to the success of this optimization problem. On the other hand, the convergence of MAML goes slower as σ increases, indicating that task adaptation becomes more difficult for task distributions that are more complex. In Figure 3.2, we follow similar protocols to the experiments conducted for MaxCut, but switch the base matrix to Sherrington-Kirkpatrick model, and its geometrically local 1D and 2D versions. Similar results are observed for these problems, where MAML can discover solutions that outperform random initialization using far fewer iterations with SGD, and the performance is competitive with that of stochastic reconfiguration training from scratch. This implies that MAML is robust with respect to the choice of Hamiltonian type. In Figure 3.3, we switch the architecture from RBM to convolutional neural networks that take into account the geometric information of the lattice. The advantage of MAML persists in disordered but geometrically local environments, and MAML performance is largely on par with stochastic reconfiguration. This experiment indicates that MAML is robust with respect to both the choice of the model architecture.

We conclude from our experiments that single-task learners are consistently slow to converge and the convergence remains slow when using the stochastic reconfiguration method. In some cases, for example, in the experiments in Figure 3.3, meta-learners achieved the same or better long-term energy as single-task learners trained with stochastic reconfiguration, at significantly decreased computational cost. In addition, foMAML and MAML exhibit overall better performance on task distribution $\mathcal{T}$ with smaller strength of disorder factor σ, in comparison with the random weight initialization counterparts. This is expected as larger σ corresponds to a sparser task distribution where the samples are less related. The improvements from MAML initializations are robust with respect to the choice of hamiltonian types and model architectures.

3.5 Conclusion

The experimental results for various matrix ensembles indicate that MAML effectively solves meta-VQMC by accelerating training and improving convergence. While the Max-Cut problem is exactly solvable for the graph sizes considered here (*e.g.*, by the Branch and Bound method [108]), the experiment illuminates the importance of Riemannian geometry to the success of the algorithm in some learning environments. It would be interesting to investigate if similar findings impact the conclusions of policy-gradient-based Meta-RL. In the case of quantum spin systems, the advantage offered by SR optimization is less significant, while MAML maintains the lead compared to SGD both in terms of acceleration and convergence. The reduction in performance with the increasing disorder is expected

on general grounds since MAML has less opportunity to exploit regularities in the task distribution. We speculate that the geometrically local models provide further opportunities to exploit task regularity, explaining the improved convergence MAML relative to SR in these environments. The models investigated all have non-positive off-diagonal entries, which is a simplifying assumption and can be relaxed, paving the way to investigating matrix ensembles relevant to electronic structure. The ideas presented in this chapter naturally extend also to variational quantum algorithms (VQAs) such as the variational quantum eigensolver. The key difference in the case of VQAs is that the denominator in the Rayleigh quotient (2.3) is normalized $\langle \psi_\theta, \psi_\theta \rangle = 1$ and stochastic estimation of the quantum expectation value $\langle \psi_\theta, H_\tau \psi_\theta \rangle$ involves performing measurements in multiple bases if the Hamiltonian contains non-commuting terms. The exploration of meta-VQA and associated learning algorithms is left to future work.

Chapter 4

Accelerating VQMC with Normalizing Flows

The fact that the state space of a quantum system scales exponentially with the number of its constituents leads to an inevitable curse-of-dimensionality facing the exact simulation of generic quantum many-body systems. In practice, approximate solutions are sufficient for most purposes and a number of successful variational methods based on the Rayleigh-Ritz principle have been developed, which, given a local Hamiltonian H, produce an estimate for the minimal eigenvalue $\lambda_{\min}(H)$ and a description of an associated eigenvector. Nevertheless, complexity-theoretic arguments suggest that the curse-of-dimensionality is ultimately unavoidable [1] and the investigation of scalable variational algorithms is an active field of research. A particularly promising variational algorithm from the viewpoint of scalability is the variational quantum Monte Carlo (VQMC) [88].

VQMC targets the ground eigenstate by performing alternating steps of Monte Carlo sampling from a high-dimensional quantum state followed by gradient-based optimization. By exploiting neural networks as trial wavefunctions, Carleo and Troyer [24] showed that VQMC can achieve state-of-the-art results for the ground state energies of physically interesting magnetic spin models. Unfortunately, the increased flexibility afforded by neural networks comes at the cost of rendering exact Monte Carlo sampling intractable, which necessitates the use of a Markov chain Monte Carlo (MCMC) sampling strategy.

However, MCMC sampling limits the scalability of VQMC in two ways: (1) the burn-in process is an inherently sequential task; (2) sampling precise and uncorrelated samples become increasingly difficult for large input dimension. Autoregressive models, in contrast, provide efficient and exact computations for both sampling and density evaluation that are GPU-supported. Recently, autoregressive neural quantum states have been introduced [120], which has allowed the VQMC to enjoy the advantages that autoregressive models have previously provided in machine learning. Inspired by the ability of autoregressive models to eliminate the reliance of the VQMC on the MCMC, we undertake a parallelization study of autoregressive neural quantum states, thereby improving the time-efficiency and scalability of VQMC.

4.1 Background

The idea of utilizing neural network quantum states to overcome the curse of dimensionality in high-dimensional VQMC simulations was first introduced by Carleo and Troyer [24], who concentrated on restricted Boltzmann machines (RBMs) applied to two-dimensional quantum spin models. Sharir *et al.* [120, 119] introduced neural network quantum states based on the autoregressive assumption inspired by PixelCNN [136] and demonstrated significant improvement in performance compared to RBMs. The autoregressive assumption was subsequently explored in VQMC using recurrent neural wavefunctions [60]. Autoregressive models have also been used to solve statistical mechanics models in [142]. Since our focus is on the scalability of VQMC, particularly in situations where MCMC is expected to struggle, unlike [24, 120, 60] we consider non-geometrically local Hamiltonians without an underlying lattice structure. This also contrasts with the work of [91], who considered parallelization of VQMC using MCMC sampling but assuming geometric locality. It was recently shown [47, 150] that techniques from quantum VQMC literature [24] can be adapted for approximately solving combinatorial optimization problems.

We explore the scalability of VQMC when autoregressive models [13], with exact sampling, is used in place of MCMC. Autoregressive models estimate the joint distribution by decomposing it into a product of conditionals by the probability chain rule, making both the density estimation and generation process tractable. To this end, Larochelle and Murray [75] proposed neural autoregressive distribution estimator (NADE) as feed-forward architectures. MADE [43] improves the efficiency of models with minor additional costs for simple masking operations. Sharir *et al.* [120, 119] introduced neural-network quantum states based on the autoregressive assumption inspired by PixelCNN [136] and demonstrated significant improvement in performance compared to RBMs.

Larochelle and Murray [75] proposed neural autoregressive distribution estimator (NADE) as feed-forward architectures. MADE [43] improves the efficiency of models with minor additional costs for simple masking operations. For probabilistic generative models, unnormalized models such as RBM rely on approximate sampling procedures like MCMC, whose convergence time remains undetermined, which often results in the generation of highly correlated samples and deterioration in performance. Such sampling approximations can be avoided by using autoregressive models [13] that estimate the joint distribution by decomposing it into a product of conditionals by the probability chain rule, making both the density estimation and generation process tractable. Kingma et al. [69] used autoregressive models as a form of normalizing flow [70].

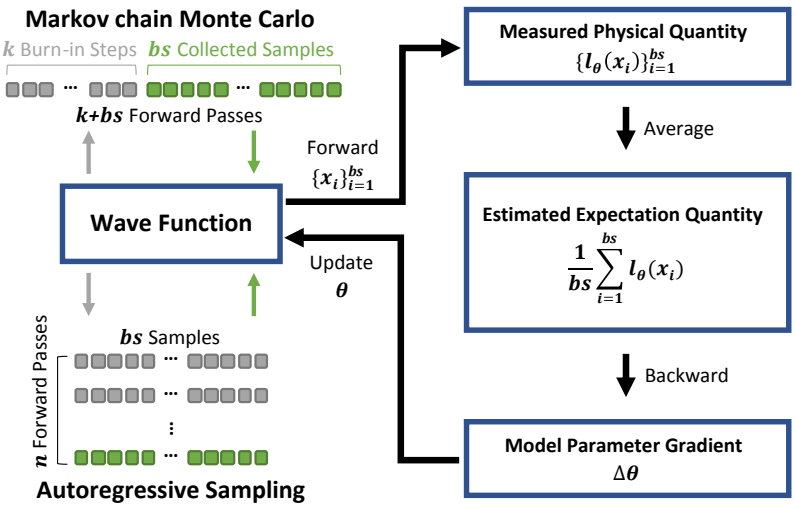


Figure 4.1: Overview of our algorithms, with illustrations of the comparison between Markov chain Monte Carlo sampling (MCMC) and autoregressive sampling (AUTO) on the left, and the VQMC optimization procedure on the right. MCMC sampling involves $k + bs/c$ forward passes, where k is the number of burn-in samples, c is the number of sampling chains ($c = 1$ in the figure) and bs is the batch size; AUTO only requires n forward passes to sample *exactly* from the distribution of interest.

4.1.1 Autoregressive Models

Now we discuss the modeling assumptions which enforce normaliztion of the differentiable trial function $\psi_\theta : [N] \to \mathbb{R}$, and thus eliminate the need for MCMC sampling. An elegant method to impose normalization is to make use of an autoregressive assumption, which has recently been generalized to neural network quantum states in [120, 60]. Since we are targeting a ground eigenvector, which is known to be non-negative, we may assume without loss of generality that $\psi_\theta(x) = \sqrt{\pi_\theta(x)}$, thereby shifting the modeling assumption into the choice of a normalized distribution π_θ satisfying the following condition,

$$\pi_\theta(x) = \prod_{i=1}^{n} \pi_i(x_i | x_{i-1}, \ldots, x_1) \quad . \tag{4.1}$$

Many proposals for neural networks satisfying the autoregressive assumption have been put forth. In this work we follow Germain et al. [43], who proposed the masked autoencoder for distribution estimation (MADE) which computes all conditionals in one forward pass using a single network with appropriate masks. Recall that a single hidden layer autoencoder is described by the following composition of functions,

$$g_1(x) = \max\{0, W_1 x + b_1\} \tag{4.2}$$

$$g_2(x) = \sigma(W_2 g_1(x) + b_2) \quad , \tag{4.3}$$

Algorithm 3 Autoregressive Sampling [43] (Batch Size 1)

Input: Randomly initialized state x^0 of size n
Output: Sampled state x^* of size n
for i-th out of n iterations **do**
  Compute $p(x_i|x_{1:i-1}^{i-1})$ with a forward pass
  Sample $y_i \in \{\pm 1\}$ with $p(\cdot|x_{1:i-1}^{i-1})$
  Get x^i by updating x_i^{i-1} with y_i
end for
Set $x^* = x^n$

and where the rectification and sigmoid functions are applied elementwise. MADE achieves the desired autoregressive assumption by appropriate application of binary masks M_1 and M_2 to the weight matrices defining the autoencoder, resulting in a MADE layer of the form

$$\begin{aligned} g_1(x) &= \max\{0, (M_1 \odot W_1)x + b_1\} \\ g_2(x) &= \sigma\left((M_2 \odot W_2)g_1(x) + b_2\right) \quad , \end{aligned} \tag{4.4}$$

where $\odot$ denotes elementwise multiplication.

In Figure 4.1, we compare the sampling procedures between MCMC and AUTO (as described in Algorithm 3). MCMC involves $k + bs/c$ forward passes, where c is the number of sampling chains and *bs* is the batch size. Although the number of forward passes can be reduced by increasing the number of chains, the number of burn-in iterations k required for convergence is undetermined and cannot be parallelized. On the other hand, AUTO only requires n forward passes to sample exactly from the distribution of interest.

4.1.2 Quantum Hamiltonians and QUBO Problems

In this section, we consider a family of matrices motivated by quantum physics, which are parametrized by $O(\text{poly}(n))$ real parameters $\alpha_i, \beta_i, \beta_{ij} \in \mathbb{R}$ as follows,

$$H = -\sum_{1 \le i \le n} \left(\alpha_i X_i + \beta_i Z_i\right) - \sum_{1 \le i < j \le n} \beta_{ij} Z_i Z_j, \tag{4.5}$$

where $X_i := I^{\otimes(i-1)} \otimes X \otimes I^{\otimes(n-i)}$ and $Z_i := I^{\otimes(i-1)} \otimes Z \otimes I^{\otimes(n-i)}$ are $2^n \times 2^n$ matrices defined in terms of the following elementary 2×2 matrices,

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad , \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad , \qquad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad . \tag{4.6}$$

It is easily verified that H meets the conditions of definition 2.2.1 with sparsity parameter $s = n$. In terms of the binary representation of the row index $x = 2^{n-1}x_1 \cdots 2^0 x_n$ and the column index $y = 2^{n-1}y_1 \cdots 2^0 y_n$, the matrix entries of H are given by

$$\begin{aligned} H_{xy} = &- \sum_{1 \leq i \leq n} \left(\alpha_i \delta_{x_1 y_1} \cdots \delta_{\neg x_i y_i} \cdots \delta_{x_n y_n} + \beta_i (1 - 2x_i) \right) \\ &- \delta_{xy} \sum_{1 \leq i \leq j \leq n} \beta_{ij} (1 - 2x_i)(1 - 2x_j) \end{aligned} \tag{4.7}$$

and $\neg x_i$ denotes logical negation of $x_i \in \{0, 1\}$. For simplicity we imposed $\alpha_i \geq 0$ to ensure that the ground eigenvector can be chosen to be a non-negative vector as a consequence of the Perron-Frobenius theorem.

In the special case where $\alpha_i = \beta_i = 0$ and $\beta_{ij} = \frac{1}{4} L_{ij}$ where L is the adjacency matrix of an undirected graph $G = (V, E)$ of size $|V| = n$, the ground state problem coincides with the Max-Cut problem, and thus VQMC can be employed as a heuristic for approximate combinatorial optimization [47, 150], which is equivalent to natural evolution strategies [150].

4.2 Parallelization

Unlike standard Monte Carlo methods, MCMC cannot be parallelized easily. The fundamental limitation is easily seen: to generate a sample x_{t+1} from a Markov chain, we need to sample the transition kernel $p(\cdot|x_t)$, which requires knowledge of the immediate past state x_t. This sequential nature of the sampling immediately precludes any direct attempt at parallelizing the sampling process.

We could attempt to initialize multiple independent sampling chains; indeed, this is one of the standard approaches often implemented in Bayesian inference frameworks. But when sampling a high-dimensional distribution using a random walk Metropolis-Hastings, it typically takes a very long time for the random walk to explore the parameter space. This significantly slows down the convergence of the estimates (2.9) to the true expectation value; furthermore, it is very difficult to determine *a priori* how many samples will be required for this convergence within a specified tolerance. In practice, MCMC first discards a pre-determined number of samples in each of the independent chains to avoid the transient Markov transitions (a.k.a. burn-in) and down-samples the remainder by selecting samples at regular intervals to reduce correlations (a.k.a. thinning). Any expectations are then computed based on this smaller set of selected samples. Improper choice of these parameters can severely degrade the quality of the generated estimates. Furthermore, they also reduce the

parallel efficiency; suppose k samples are discarded as burn-in and every j-th sample is selected during thinning. Then constructing n samples on each of L independent computing units will lead to a parallel efficiency of

$$\frac{k+(nL-1)j+1}{k+(n-1)j+1} = 1 + \frac{nj}{k+(n-1)j+1}(L-1) = a + bL \tag{4.8}$$

for some a and b depending on k, j and n. Note that this calculation is solely focused on the sampling task, and therefore does not take into account any communications that might be necessary between the computing units for obtaining the final result. Even then, as the number of burn-in samples k is increased, the slope b decays from 1 towards 0 ($b = 1$ is indicative of optimal scaling).

On the other hand, an autoregressive model (AUTO) can generate exact samples from the target distribution. Although the implementation of AUTO has a sequential nature that scales linearly with the input dimension, it can generate independent samples from the target distribution by transforming *i.i.d.* samples from a simple distribution (*e.g.*Gaussian). This step is easily parallelized: as long as we have identical copies of the autoregressive model in a number of computing units (*e.g.*GPUs), we can construct independent samples in parallel. Communication between the computing units is necessary only when we need to update the parameters of the neural network, *e.g.*during a stochastic gradient descent update.

Our model consists of fully connected weight matrices; therefore as we scale up the problem size, the bottleneck for our algorithm is the memory usage. For example, assuming a GPU can only store models with up to 10M parameters, we can set the size of the hidden layer to 500 at maximum when solving a problem with 10K input dimensions. This limitation can be addressed along with two complementary but independent avenues:

1. **Model Parallelization:** Distribute the model parameters across computing units, so that each unit needs to store and update a small part of the model.

2. **Sampling Parallelization:** Use identical copies of the model across the computing units to generate only a few samples per unit, and combine the independent samples from all these units to construct an accurate expectation estimate.

The communication pattern between the computing units in model parallelization is intimately linked with the choice of the autoregressive neural network while the sampling parallelization is model agnostic.

In this work, we restrict our attention to only parallelizing the sampling step. Consider a quantum Hamiltonian of size $N = 2^n$ and an autoregressive model with two hidden layers of size h. Given a total number of L computing units/GPUs and a mini-batch

Table 4.1: Training time (measured in seconds) comparison on TIM for 300 training iterations with one GPU. Our MCMC settings are introduced in Section 4.3.1. The running time of MADE&AUTO scales roughly linearly with respect to the number of dimensions, due to the sequential nature of its sampling procedure, but significantly outperforms RBM&MCMC in practice.

Model	Optimizer	Sampler	# of Dimensions				
			20	50	100	200	500
RBM	ADAM	MCMC	135.64	154.25	189.91	249.40	456.68
MADE	ADAM	AUTO	2.85	5.74	10.63	20.45	49.62

size of mbs samples to be drawn on each GPU, we end up with an effective batch size of $bs = L \times mbs$. Locally, each process first generates mbs samples, then computes the physical measurements with the samples, and finally uses backpropagation to get the gradient of the model parameters. These local gradient vectors have length $d = 2hn + h + n$, which are averaged over the GPUs using a parallel reduction. Each GPU then updates its own model parameters locally.

The computation complexity can be estimated as follows: during the local sampling process on each GPU, the algorithm involves n forward passes for sampling, and a fixed number of forward passes for physical quantity measurements. The dominant cost of each forward pass is multiplication by $h \times n$ and $n \times h$ matrices, both $O(hn)$; this leads to a total computational cost of $O(hn^2 \times mbs)$ flops per GPU. Computing the average gradient over GPUs using parallel reduction costs further $O(hn)$ flops, and involves communication of $O(hn)$ floating point numbers. Clearly, the parallel efficiency is given by

$$\frac{O(hn^2 \times bs)}{O(hn^2 \times mbs) + O(hn)} = \frac{O(hn^2 \times L \times mbs)}{O(hn^2 \times mbs) + O(hn)} \tag{4.9}$$

Since the constants in the $O(hn^2 \times L \times mbs)$ and the $O(hn^2 \times mbs)$ are the same, this ratio is approximately L when n or mbs are large.

4.3 Experiments

This section contains an extensive evaluation of our approach. We first compare AUTO sampling and MCMC sampling in Section 4.3.2, where the advantage of AUTO in terms of computational efficiency becomes clear for problems of higher dimensions. The convergence performance is shown in Section 4.3.3. Our algorithm is competitive against the state-of-the-art SDP solvers for small/medium scale Max-Cut problems. In Section 4.3.4, we demonstrate the scalability of our technology by solving large-scale problems up to 10K dimensions. We achieved near-optimal weak scaling, and the convergence of our model improves as we

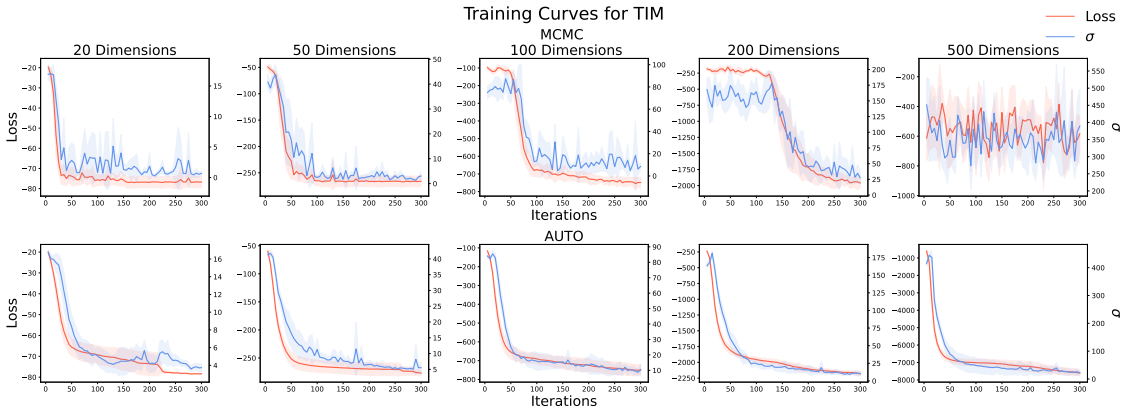


Figure 4.2: Training curves for TIM, where the red curves refer to the training loss/energy, and the blue curves refer to the standard deviation of the stochastic objective, which should be zero when the wave function converges to the exact ground-state. By fixing the learning rate and the total number of training iterations, it becomes more difficult for RBM&MCMC to converge as the problem size grows, due to the inaccurate estimation of the population energy by the low-quality MCMC samples. The training of MADE&AUTO is stable across all problems.

increase the effective training batch size.

4.3.1 Experimental Setup

In this section, we evaluate VQMC using two non-geometrically local Hamiltonians: the Max-Cut and the transverse field Ising model (TIM) model. In the case of Max-Cut, the adjacency matrix was chosen by forming the $n \times n$ matrix $(B + B^T)/2$ with $B_{ij} \sim \text{Bernoulli}(0.5)$ sampled once and fixed, followed by rounding and setting diagonal entries to zero. The second example is a disordered quantum system referred to as transverse field Ising model model, whose Hamiltonian is of the form (4.7) with $\beta_i, \beta_{ij} \sim U(-1, 1)$ and $\alpha_i \sim U(0, 1)$ sampled once and fixed.

For Max-Cut, we compare our approach against VQMC with MCMC sampling [47, 150], as well as the semidefinite programming (SDP) relaxation approximation algorithms including Goemans-Williamson Algorithm [46] and the Burer-Monteiro reformulation with the Riemannian Trust-Region method [2]. As an additional baseline, each model is also trained using the SR method. We benchmark the running time and converged energy of our model on TIM in our scalability experiments.

Model architecture

Network architecture is chosen to be MADE and is compared against RBM, proposed by Carleo and Troyer [24], taking the one-dimensional state as input and outputs the logarithmic

probability amplitude. The structure of MADE is as follows

$$\begin{aligned}\textit{Input} &\xrightarrow{[bs,n]} \texttt{MaskedFC1} \xrightarrow{[bs,h]} \texttt{ReLU} \\ &\xrightarrow{[bs,h]} \texttt{MaskedFC2} \xrightarrow{[bs,n]} \texttt{Sigmoid} \xrightarrow{[bs,n]} \textit{Output},\end{aligned}$$

and the structure of RBM is

$$\begin{aligned}\textit{Input} &\xrightarrow{[bs,n]} \texttt{FC}_{n,h} \xrightarrow{[bs,h]} \texttt{Lncoshsum} \xrightarrow{[bs]} \textit{Output}1 \\ &\xrightarrow{[bs,n]} \texttt{FC}_{n,1} \xrightarrow{[bs]} \texttt{Add}\ \textit{Output}1 \xrightarrow{[bs]} \textit{Output}.\end{aligned}$$

Here *bs* is the batch size and n is the number of dimensions. $\texttt{FC}_{a,b}$ is a fully connected layer with input size a and output size b; and `MaskedFC` is the masked version of `FC`, to remove the connections in the computational path of MADE. `Lncoshsum` refers to a series of linear and non-linear operations involving: 1) taking natural logarithm for each entry of the input tensor; 2) taking hyperbolic cosine for each entry of the input tensor; 3) summation over the last dimension of the input tensor. The size of the tensor being passed to the next operator is indicated above the right arrows.

For large-scale problems with high dimensional input size n, we need to choose a proper latent size h to balance between the memory usage and the capacity of the model. In our experiments, we set $h = 5(\log n)^2$ as the hidden layer size for MADE and $h = n$ as the number of hidden units for RBM.

Training. All models are trained for 300 iterations. In our single-GPU experiments, at each iteration, the model is updated with a batch of 1024 training samples. For evaluation, we draw a batch of 1024 testing samples from trained model, and report their mean energy. Two base optimizers are considered: stochastic gradient descent (SGD) with learning rate 0.1 or ADAM with learning rate 0.01, where the latter is our default optimizer. In addition, we provide additional results on models trained using the SR [124] method for performance comparison. The SR optimization was performed using a regularization parameter $\lambda = 0.001$ and a learning rate 0.1. No learning rate scheduler is applied. For scalability experiments, each GPU is distributed with a constant mini-batch size mbs, and the effective batch size is $mbs \times L$, where L is the total number of GPUs available.

Our MCMC sampler is the random walk Metropolis–Hastings algorithm, running with two chains. We expect that it takes more effort for MCMC to converge for large-scale problems. Therefore, for each chain, we set heuristically the burn-in iterations k to scale linearly with respect to the input dimension n, *i.e.*, $k = 3n + 100$.

Throughout the experiments, the timing benchmarks are performed on NVIDIA Tesla

Table 4.2: Optimized objective (maximize cut number for Max-Cut, minimize ground state energy for TIM) values for different problem sizes and different optimizers, averaged over 5 runs with different random seeds. The first three rows in the Max-Cut section consist of results from running classical algorithms and serve as benchmarks. For the rest of the rows in the table, the batch size is fixed to be 1024. We note that MADE&AUTO achieves satisfactory performance in the sense that it's directly comparable with the SDP solvers on Max-Cut. On the other hand, RBM&MCMC takes longer to converge as the problem size grows, whereas the convergence of MADE&AUTO remains stable.

Problem	Model	Sampler	Optimizer	# of Dimensions				
				20	50	100	200	500
Max-Cut	Classical: Random			27.2 ± 2.2	150.4 ± 5.8	610.4 ± 11.6	2495.8 ± 42.8	15696.0 ± 16.8
	Classical: Goemans-Williamson			41.4 ± 2.0	194.2 ± 2.3	741.0 ± 11.1	2881.6 ± 14.4	17242.4 ± 37.3
	Classical: Burer–Monteiro			**43.0** ± 0.0	**200.0** ± 0.0	754.0 ± 3.0	**2928.0** ± 3.7	**17416.0** ± 23.13
	RBM	MCMC	SGD	41.4 ± 1.5	192.0 ± 3.3	733.8 ± 13.0	2825.6 ± 5.5	15945.6 ± 44.2
			ADAM	40.6 ± 1.6	190.2± 2.7	719.8 ± 6.6	2777.6 ± 14.2	16576.0 ± 30.9
			SGD+SR	**43.0** ± 0.0	198.8 ± 1.5	758.0 ± 1.1	2898.0 ± 22.0	15956.8 ± 29.9
	MADE	AUTO	SGD	42.6 ± 0.4	192.0 ± 2.4	742.2 ± 5.9	2846.0 ± 4.8	16880.0 ± 73.6
			ADAM	42.4 ± 0.8	193.8 ± 3.1	733.8 ± 9.1	2847.8 ± 12.1	17006.6 ± 23.0
			SGD+SR	**43.0** ± 0.0	**200.0** ± 1.5	**758.4** ± 6.5	2909.2 ± 3.1	17176.6 ± 30.5
TIM	MCMC	RBM	SGD	-80.22 ± 2.79	-270.65 ± 9.64	-762.11 ± 28.58	-1981.17 ± 72.19	-976.25 ± 119.43
			ADAM	-80.38 ± 2.42	-265.47 ± 8.21	-756.33 ± 16.73	-2216.45 ± 31.95	-924.53 ± 121.10
			SGD+SR	-80.70 ± 2.10	**-282.02** ± 8.37	-764.74 ± 14.67	-2234.23 ± 36.72	-1046.40 ± 334.50
	MADE	AUTO	SGD	-80.30 ± 0.01	-281.18 ± 5.51	-767.88 ± 13.45	-1872.16 ± 41.89	-6773.97 ± 233.19
			ADAM	-80.48 ± 0.18	-277.11 ± 4.48	-771.11 ± 17.06	-2181.31 ± 33.39	-7597.37 ± 171.25
			SGD+SR	**-81.25** ± 0.07	-277.23 ± 9.96	**-812.33** ± 12.55	**-2252.12** ± 84.00	**-8673.27** ± 304.45

V100 GPUs, with 32GB of memory for each.

4.3.2 MCMC vs. AUTO: Runtime

Despite the sequential nature of both MCMC and AUTO sampling, in practice, AUTO sampling can be operated with GPU in a straightforward fashion and exhibit superior running time efficiency. Our results on the running time comparison is shown in Table 6.2.

The running time of RBM&MCMC scales with the total number of iterations in each chain, which includes a fixed number of burn-in iterations that cannot be parallelized. In our setting, we set the number of chains to be 2, and burn-in iterations k that grows linearly with respect to the input dimension n. In principle, the running time of MCMC can be reduced further by increasing the number of chains or choosing a smaller k. However, a more severe problem of MCMC lies in the fact that the distribution of the samples generated by MCMC only converges to the distribution of interest asymptotically. As the input dimension increases, it becomes more difficult for the random walk Metropolis–Hastings algorithm to converge, which can potentially affect the quality of generated samples if k is not properly chosen. The running time of MADE&AUTO is dominated by the sampling time that scales linearly with respect to the input dimension n, which significantly outperforms its RBM&MCMC counterpart. More importantly, for AUTO, we know exactly the

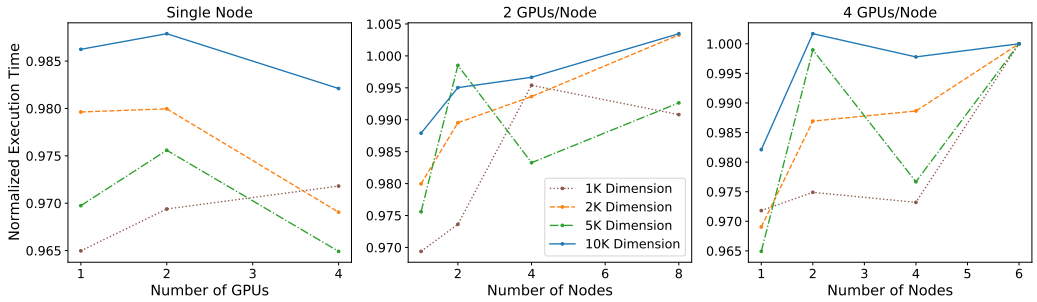


Figure 4.3: Sampling times for the TIM problem in 1K, 2K, 5K and 10K dimensions with mini-batch sizes *mbs* = 512, 128, 16 and 4 samples per GPU, respectively. The minibatch sizes were chosen to saturate GPU memory per problem dimension. All times are normalized by the execution time of the largest GPU configuration (6×4) for each dimension. Note that the normalized executions times are all close to 1, indicative of near-optimal weak scaling.

computational complexity needed to get correct samples from the distribution of interest, as opposed to MCMC that requires undetermined number of iterations to converge.

The corresponding training curves are shown in Figure 4.2, where the red curves refer to the training loss/energy, and the blue curves refer to the standard deviation of the stochastic objective, which approaches zero as the wave function converges to the exact ground-state. RBM&MCMC converges reasonably well on small-scale problems, but has more difficulty to converge as the problem scales up. On the other hand, our model converges rapidly and stably to low energy across problems of different scales. This observation motivates us to attempt to solve problems of even higher dimensions.

4.3.3 MCMC vs. AUTO: Convergence Study

The convergence result of our model on the Max-Cut problems is shown in Table 4.2, where we compare MADE&AUTO against the state-of-the-art SDP relaxation approximation algorithms developed in the past decades, as well as VQMC with RBM&MCMC.

Random Cut algorithm is a simple randomized 0.5-approximation algorithm that randomly assigns each node to a partition. Goemans and Williamson [46] improved the performance ratio from 0.5 to at least 0.87856, by making use of the semidefinite programming (SDP) relaxation of the original integer quadratic program. Burer and Monteiro [22] reformulated the SDP for Max-Cut into a non-convex problem, with the benefit of having a lower dimension and no conic constraint. The implementation of Goemans-Williamson Algorithm used the `CVXPY` [30, 3] package and the Burer-Monteiro reformulation with the Riemannian Trust-Region method [2] used `Manopt` toolbox [16], which essentially implements the optimization algorithm proposed by [65].

For evaluation, we constructed a problem instance for each Hamiltonian size $n \in$

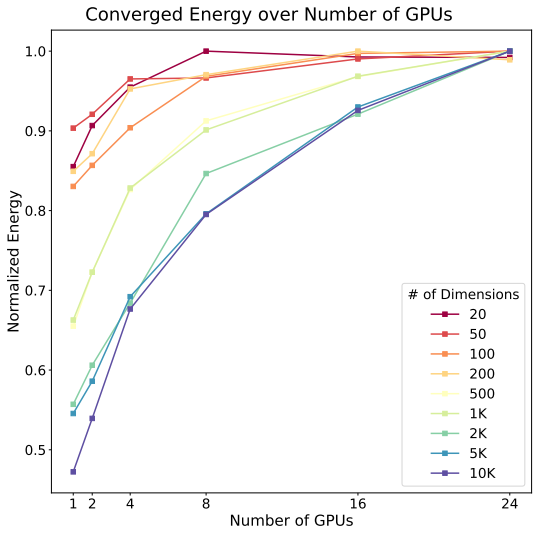


Figure 4.4: Normalized converged energy for TIM problems of different sizes. Each GPU is distributed a batch size of 4; the total effective batch size equals 4 times the total number of GPUs used. The energy is normalized for each problem size (the values from each curve are divided by the one with the largest magnitude among them). The converged energy improves as the total number of GPUs (effective batch size) increases. The improvement saturates for smaller problems, which also implies that a larger batch size is required for larger problems.

$\{20, 50, 100, 200, 500\}$ by randomly generating parameters defined in Eq. 4.5. For each problem instance, each algorithm was executed 5 times using 5 random seeds. In Table 4.2, we report the averaged result over problem instances of different sizes.

In general, MADE&AUTO slightly outperforms RBM&MCMC on small-scale problems, and the latter fails to converge for problems of input dimension 500, due to our constraint on the number of training iterations.

The natural gradient descent [5, 124] proved essential for converging to a good local optimum. We apply the SR to both VQMC methods and observe similar improvements: optimizers equipped with SR are consistently improved over all architectures. On the other hand, the performance of our algorithm with SR is competitive against the state-of-the-art SDP solvers on Max-Cut problems.

4.3.4 AUTO: Multi-GPU Scalability

By distributing the sampling task across multiple GPUs, our method can extend to large-scale problems (with input dimensions up to 10K) by reducing the mini-batch size *mbs* distributed to each GPU. The effective batch size depends on both *mbs* and the number of GPUs L available for training.

In Figure 4.3, we plot the normalized execution times for the 1K, 5K and 10K dimen-

sional TIM problems as we vary the number of GPUs and the GPU distribution across nodes. We choose the minibatch sizes assigned to each GPU depending on the dimensionality of the problem so that the GPU memory is saturated. Note that for both intra-node and inter-node distributed sampling schemes, the execution times remain nearly constant as long as the number of samples per GPU is kept fixed. This is indicative of near-optimal weak scaling: consider a problem so large that we are able to generate only a few samples using a single GPU due to memory constraints. In this scenario, by using a large number of GPUs to generate independent sets of samples, we should be able to drive the stochastic optimization problem to convergence.

The effective batch size increases as we scale up the number of GPUs. This improves the convergence performance of our method. We benchmark the result in Figure 4.4, where we train our models across different numbers of GPUs, on TIM problems of different sizes. The improvement saturates for smaller problems as the effective batch size increases but remains significant for larger problems. This implies that our model requires a larger batch size to achieve optimal performance for problems of a larger scale. Intuitively, batch size quantifies the exploration capability in the state space: the algorithm has a better chance to discover the ground state if it is allowed to explore more.

The raw data of our experiments in this section is provided in Section 4.4.4.

4.4 Case Studies

In this section, we conduct experiments on several aspects of our settings in more detail, to support our conclusions that MADE+AUTO significantly outperforms RBM+MCMC in terms of the convergence rates for large-scale problems. Throughout this section, we train our models for Max-Cut problems with ADAM optimizer on a single GPU. All results are averaged over 5 runs with different random seeds.

4.4.1 Ablation Study: Latent Size

We conduct ablation studies on the choice of latent size for our models. Latent size refers to the number of hidden units for RBM and the hidden layer size for MADE.

In Table 4.3, we train both MADE and RBM on Max-Cut problems with graph sizes $n \in \{50, 100, 200, 500\}$ under different choices of latent size $h \in \{(\log n)^2, 3(\log n)^2, n, 5n, n^2\}$. We also cite the numbers from Table 4.2 for direct comparison, where we adopt $h = 5(\log n)^2, n$ for MADE and RBM, respectively. We measure the training time of each model for 300 iterations in seconds and present the numbers on the right side of the table. The

Table 4.3: Ablation study on the latent size. We train the models with ADAM on Max-Cut problems. n is the graph size. Optimal performance is obtained under a proper choice of latent size h; MADE falls off if we push GPU to its computational limits.

Model	n	Latent size h										
		Cut table						Time table				
		$(\log n)^2$	$5(\log n)^2$	n	n	$5n$	n^2	$(\log n)^2$	$3(\log n)^2$	n	$5n$	n^2
MADE	50	191.	193.8	-	195.	194.6	195.	7.22	7.19	7.24	7.42	7.41
	100	735.8	733.8	-	734.2	731.2	726.2	13.43	13.49	13.48	13.90	13.96
	200	2832.8	2847.8	-	2848.6	2821.4	2779.	26.49	25.78	26.07	26.85	57.19
	500	16905.4	17006.6	-	16973.8	16872.8	16311.4	64.81	66.48	67.79	105.97	1426.92
RBM	50	193.	-	190.2	192.	192.2	191.4	151.07	151.49	150.72	150.71	152.68
	100	721.	-	719.8	730.2	711.	705.2	181.11	180.30	180.47	182.15	183.62
	200	2786.2	-	2777.6	2779.6	2765.6	2747.4	242.95	241.05	243.24	243.91	246.05
	500	16568.8	-	16576.0	16652.6	16577.2	16543.	427.23	429.07	432.39	428.17	510.02

Table 4.4: Ablation study on the MCMC sampling scheme. We train the RBM with ADAM on Max-Cut problems. n is the graph size; $\{n, 10n\}$ and $\{\times 2, \times 5, \times 10\}$ are from Scheme 1 and Scheme 2, respectively.

Model	n	Sampling scheme										
		Cut table						Time table				
		n	$3n$+100	$10n$	×2	×5	×10	n	$10n$	×2	×5	×10
MCMC	50	190.8	190.2	193.8	191.6	192.6	192.8	110.44	197.02	199.64	500.02	1004.96
	100	700.2	719.8	733.	706.8	720.	729.8	124.01	296.83	201.52	507.65	1011.51
	200	2674.8	2777.6	2795.4	2670.4	2720.6	2736.8	143.76	492.31	206.91	514.80	1023.43
	500	16205.	16576.0	16626.6	16022.2	16066.6	16156.6	212.86	1103.18	207.43	508.43	1021.21

results are averaged over 5 runs with different random seeds.

Several observations can be made. First, optimal performance is obtained under a reasonable choice of h, between $3(\log n)^2$ and n; models with a latent size that is either too large or too small do not perform well. Second, the time complexity usually does not scale with the model size when running on GPU. However, MADE falls off if we push GPU to its computational limits, *e.g.*, AUTO sampling $bs = 1024$ samples from MADE with $O(n^3)$ parameters. This is in practice not a serious concern for MADE with latent size $h = O((\log n)^2)$ as it will always face its memory bottlenecks first by storing the batch of high dimensional inputs as the problem size increases. Third, we re-did the experiments on RBM with n hidden units and obtain slightly different results in Table 4.2, due to different choices of random seeds and machines that the model is trained on.

4.4.2 Ablation Study: MCMC Sampling Scheme

We conduct ablation studies on the choice of MCMC sampling schemes. In particular, we consider:

- Scheme 1: the sampler discard the first $\{n, 10n\}$ samples in the chain and keep the next *bs* samples.

Table 4.5: Time elapsed to reach the target performance, measured in seconds. We train the RBM with ADAM on Max-Cut problems. At every iteration, after the training updates, we sample another batch of samples for evaluation; the algorithm terminates if the evaluation score surpasses the target score. Evaluation time is not taken into account.

Method	# of Dimensions (Targeted cut number)				
	20(41)	50(190)	100(730)	200(2800)	500(16800)
MADE+AUTO	3.14	3.61	20.08	3.25	6.27
RBM+MCMC	126.84	154.09	247.91	612.76	1096.08

- Scheme 2: the sampler takes every $\{2, 5, 10\}$th sample in the chain until *bs* samples are collected in total.

In Table 4.4, we train RBM on Max-Cut problems with graph sizes $n \in \{50, 100, 200, 500\}$ under different choices of MCMC sampling schemes $\{n, 10n, \times 2, \times 5, \times 10\}$. We also cite the numbers from Table 4.2 for direct comparison, where we discard the first k=$3n$+100 samples in the MCMC chain. We measure the training time of each model for 300 iterations in seconds and present the numbers on the right side of the table. The results are averaged over 5 runs with different random seeds.

Several observations can be made. First, schemes $10n$ or $\times 10$ with longer MCMC chains result in better performance, at the cost of longer running time. Second, when running with GPU, the time complexity only scales with the length of the MCMC chain, but not the $O(n^2)$ model size.

4.4.3 Comparison of Hitting Time

In addition to showing the running time with a fixed number of iterations in Table 6.2, we demonstrate that MADE+AUTO also significantly out-performs RBM+MCMC in the sense that the former reach a target performance faster.

In Table 4.5, we train MADE and RBM on Max-Cut problems with graph sizes $n \in \{50, 100, 200, 500\}$ with target performance $\{41, 190, 730, 2800, 16800\}$ that are heuristically chosen based on the results in Table 4.2. The performance is measured in seconds and the results are averaged over 5 runs with different random seeds. RBM+MCMC requires a significantly longer time to converge to a target performance for large-scale problems.

4.4.4 Raw Data from Multi-GPU Scalability Experiments

We distribute the sampling task across multiple GPUs, our method can extend to large-scale problems with input dimensions up to 10K dimensions, by reducing the mini-batch size *mbs*

Table 4.6: Converged energy and running time for TIM problems of different dimensions. Each GPU is distributed with a batch size of 4; the total batch size equals to 4 times the total number of GPUs used. Paralleling experiments are done across different GPU configurations, where $L_1 \times L_2$ refers to a total L_1 number of nodes with L_2 GPUs in each node, and the a total number of GPUs is $L = L_1 \times L_2$. The converged energy improves as the batch size (total number of GPUs) increases.

# GPUs	Metric	# of Dimensions								
		20	50	100	200	500	1000	2000	5000	10000
1×1	Energy	-69.64	-225.53	-656.91	-1511.22	-3862.86	-9642.54	-21962.55	-56337.84	-89733.83
	Time (s)	2.85	5.74	10.63	20.45	49.62	98.01	204.18	514.14	1067.56
1×2	Energy	-70.59	-260.91	-626.55	-1788.10	-4666.89	-12056.95	-24274.07	-73938.23	-142214.93
	Time (s)	3.06	6.00	10.81	20.36	49.47	97.29	200.32	512.39	1065.71
1×4	Energy	-82.79	-257.26	-702.94	-1778.35	-5587.58	-13797.55	-29219.47	-79650.12	-165364.75
	Time (s)	3.14	6.13	10.90	20.95	49.33	98.22	202.02	507.40	1066.03
2×2	Energy	-82.79	-257.26	-702.94	-1778.35	-5418.66	-13286.22	-28886.57	-74508.23	-159416.64
	Time (s)	3.29	6.16	10.81	20.63	49.59	98.01	204.90	512.80	1068.00
2×4	Energy	-81.49	-261.31	-766.29	-1984.61	-5886.93	-14826.83	-31665.81	-94311.98	-190800.37
	Time (s)	5.26	7.91	11.10	20.68	49.95	100.95	206.12	515.03	1085.33
4×2	Energy	-81.49	-261.31	-766.29	-1929.95	-5834.87	-14464.15	-33929.40	-93814.81	-200729.03
	Time (s)	3.55	6.22	10.92	20.60	49.86	97.98	202.73	513.87	1075.07
4×4	Energy	-81.70	-261.91	-776.00	-1892.16	-6348.56	-15636.99	-44506.68	-111165.27	-229567.37
	Time (s)	3.25	6.14	13.44	21.15	49.43	98.11	203.58	514.16	1068.51
8×2	Energy	-81.70	-261.89	-776.00	-1892.15	-5975.69	-15928.98	-46415.26	-120381.78	-224738.12
	Time (s)	3.30	6.18	10.88	20.77	49.97	98.29	203.80	520.13	1072.32
6×4	Energy	-80.99	-276.52	-769.72	-1950.40	-6672.37	-17105.77	-38496.40	-127652.29	-261517.21
	Time (s)	3.22	6.22	11.14	21.12	50.43	101.30	206.36	521.97	1067.83

distributed to each GPU. The effective batch size depends on both *mbs* and the number of GPUs L available for training. Here, we provide the raw data for our distributed computing experiments in Section 4.3.4.

In Table 4.6, we show the converged energy and running time for TIM problems of different dimensions. Each GPU is distributed with a batch size of 4; the total batch size equals to 4 times the total number of GPUs used. A number of different GPU configurations were used; $L_1 \times L_2$ indicates L_1 nodes with L_2 GPUs per node were utilized. The converged energy improves as the batch size (total number of GPUs) increases.

In Table 4.7, we show the running time (seconds) for TIM problems of different dimensions. Different from the experiments in Table 4.6, each GPU is distributed with the maximum number of batch size that can be accommodated on its memory. We note that for each dimension, the run times remain constant even as we increase the number of GPUs, increasing the effective batch size. This is indicative of near-optimal weak scaling.

Table 4.7: Running time (seconds) for TIM problems of different dimensions. Each GPU is distributed with the maximum number of batchsize that can be accommodated on its memory. A number of differnt GPU configurations were used; $L_1 \times L_2$ indicates L_1 nodes with L_2 GPUs per node were utilized. We note that for each dimension, the run times remain constant even as we increase the number of GPUs, increasing the effective batch size. This is indicative of near-optimal weak scaling.

# GPUs	# of Dimensions								
	20	50	100	200	500	1000	2000	5000	10000
	# of Samples per GPU								
	2^{19}	2^{17}	2^{15}	2^{13}	2^{11}	2^{9}	2^{7}	2^{4}	2^{2}
1×1	77.34	73.34	62.70	62.67	110.37	159.51	263.05	558.93	1058.85
1×2	76.30	73.74	62.88	62.24	110.93	160.24	263.14	562.30	1060.62
1×4	76.57	73.86	63.11	62.47	110.82	160.64	260.21	556.15	1054.41
2×2	76.24	73.82	63.02	62.56	111.20	160.94	265.71	575.51	1068.28
2×4	77.56	75.29	64.50	64.65	113.94	161.15	265.01	575.77	1075.45
4×2	76.32	73.86	63.03	62.35	111.31	164.54	266.81	566.73	1070.02
4×4	76.61	76.15	65.15	64.91	112.19	160.87	265.47	562.93	1071.24
8×2	77.01	75.13	64.59	65.27	112.46	163.78	269.40	572.13	1077.35
6×4	79.83	75.39	65.08	65.61	111.97	165.30	268.52	576.37	1073.62

4.5 Conclusion

Motivated by recent developments in VQMC made possible by autoregressive sampling, we implemented a distributed variant of VQMC and applied it to solving large-scale quantum systems for which standard random-walk Markov chain Monte Carlo sampling fails to converge. The main advantage of AUTO compared to MCMC lies in its ability to sample exactly from the distribution of interest, unlike MCMC for which the quality of the generated samples is plagued by unknown convergence time, which becomes a severe problem as the dimension of the problem increases. Empirically, we demonstrated that AUTO significantly outperforms MCMC in terms of the convergence rates for large-scale problems. Training of AUTO is also more stable than that of MCMC, finding converged solutions that are competitive against the state-of-the-art baselines for Max-Cut. The above findings motivated us to explore large-scale problems up to 10K dimensions. For that purpose, we built large models and chose a batch size to exhaust the memory usage of each GPU to be distributed. The optimality of our results is only limited by the computational resources available at hand: while the convergence performance quickly saturates for small-scale problems, it continues to improve for larger-scale problems as we scale up the number of GPUs.

Chapter 5

Application: Quantum Chemistry

Quantum many-body systems describe a vast category of physical problems at microscopic scales. In the context of ab-initio Quantum Chemistry (QC), the central topic is to unravel the quantum effects determining the structure and properties of molecules by solving the many-body time-independent Schrödinger equation describing the interaction of heavy nuclei with orbiting electrons. However, the exponential complexity with respect to the number of particles makes the analytical computations about system impractical [135].

Classical strategies discretize the problem using a finite number of basis functions, expanding the full many-body state in a basis of anti-symmetric Slater determinants. Because of the exponential scaling of the determinant space, however, exact approaches that systematically consider all electronic configurations such as the full configuration interaction (FCI) method, are typically restricted to small molecules and basis sets. Coupled cluster (CC) techniques [29, 9] are approximate methods routinely adopted in QC electronic calculations, however, the accuracy of CC is intrinsically limited in the presence of strong quantum correlations, in turn restricting the applicability of the method to regimes of relative weak correlations. Recently neural-network quantum state (NQS) methods [28, 8] have proven to be successful variational ansatz for finding the ground state of molecular systems up to 30 qubits (Li_2O). However, significant scalability challenges of the NQS approach arise when considering molecules of larger scales. The scalability issue stems from two sources, which we refer to as *local energy parallelism* and *sampling parallelism*. (*i*) The complexity of the computation of local energy scales linearly with respect to the number of terms in the molecular Hamiltonian, which induces out-of-memory (OOM) issues for larger problems. (*ii*) In order to achieve satisfactory performance, one needs to sample exact and accurate configurations from the targeted distribution, which becomes expensive or even impossible for existing approaches such as Markov chain Monte-Carlo as the dimensionality rises.

In order to address local energy parallelism for large-scale molecules, we use an efficient tensor representation of the second quantized spin Hamiltonian generated from chemical data so that the computation of the local energy is efficiently supported by GPUs. We further

employ identical copies of the model across the computing units to generate only a few samples per unit and combine the independent samples from all these units to construct an accurate expectation estimate. In addition, memory consumption is reduced through gradient accumulation, which splits the batch of samples used for training the model into several mini-batches of samples that will be run sequentially on each device under parallelization. The proposed parallelization scheme for large batch computation is particularly important for chemical molecules with a large number of terms in its electronic Hamiltonian formulation, where the calculation of local energy becomes prohibitive.

The basis of our approach to achieving sampling parallelism is our utilization of a wavefunction based on Masked Autoencoder for Distribution Estimation (MADE) [43]. Using MADE as our base model enables the development of a parallelization scheme based on [152]. Unlike restricted Boltzmann machines (RBMs) [111] and NADE [75], which have formed the basis of previous ab-initio studies, MADE is known to be lightweight and scalable to high-dimensional inputs. On the one hand, it overcomes both the asymptotical convergence issues o MCMC sampling using autoregressive sampling. On the other hand, it bypasses the inherently sequential nature of NADE with the minor additional cost of simple masking operations. In addition, we further improve the performance of our model through the autoregressive sampling of the state entries in an order that roughly matches the entanglement hierarchy among the involved qubits. Our experiments demonstrate that the proposed algorithm effectively works for molecules up to 76 qubits with millions of terms in its electric Hamiltonian.

The section is organized as follows: In section 5.1 we begin by summarizing the existing state-of-art in neural-network quantum state research as applied to ab initio quantum chemistry. Section 5.2 provides mathematical details about the nature of the Hamiltonians under consideration as well as stochastic approximation strategy based on neural networks. The proposed neural-network quantum state architecture is elaborated in section 6.2 and the parallel evaluation strategies are subsequently detailed in section 5.4. Experimental results for molecules up to 76 qubits are described in section 6.3, including abalative studies on the relevant factors controlling performance of the algorithm.

5.1 Background

Variational Monte-Carlo and Autoregressive Quantum States. The idea of utilizing neural-network quantum states to overcome the curse of dimensionality in high-dimensional VQMC simulations was first introduced by Carleo and Troyer [24], who concentrated on restricted Boltzmann machines (RBMs) applied to two-dimensional quantum spin models.

However, RBM relies on approximate sampling procedures like MCMC, whose convergence time remains undetermined, which often results in the generation of highly correlated samples and deterioration in performance. Such sampling approximations can be avoided by using autoregressive models [13] that estimate the joint distribution by decomposing it into a product of conditionals by the probability chain rule, making both the density estimation and generation process tractable. To this end, Larochelle and Murray [75] proposed neural autoregressive distribution estimator (NADE) as feed-forward architectures. MADE [43] improves the efficiency of models with minor additional costs for simple masking operations. Sharir *et al.* [120, 119] and Hibat-Allah[60] introduced neural-network quantum states based on the autoregressive assumption inspired, respectively by PixelCNN [136] and recurrent neural networks, respectively, and demonstrated significantly improved performance compared to RBMs.

Application in Quantum Chemistry. Many approximate methods specific to the QC problem [55, 74] have been discovered by researchers. The Hartree-Fock approximation treats each electron in the molecule as an independent particle that moves under the influence of the Coulomb potential due to the nuclei, and a mean-field generated by all other electrons. It calculates the expansion coefficients of the linear combination of atomic orbitals. Configuration interaction methods [121] use a linear combination of configuration state functions built from spin orbitals to restrict the active space to configuration strings. Coupled cluster approaches [29, 9] construct multi-electron wavefunctions using the exponential cluster operator to account for electron correlation, but cannot parameterize arbitrary superpositions and occasionally lead to unphysical solutions. Choo *et al.* [28] proposed an RBM-based NQS variational ansatz, leveraging the power of artificial neural networks that have recently emerged in the more general context of interacting many-body quantum matter [24]. This approach provides a compact, variational parameterization of the many-body wave function. Barrett *et al.* [8] subsequently proposed a novel autoregressive NQS architecture for ab-initio QC based on NADE [75] with hard-coded pre-and post-processing that enables exact sampling of physically viable states. These advances ultimately allow their approach to scale to systems up to 30 qubits, surpassing what was previously possible using RBMs, while still falling short of large-scale applications. Empirically, we have found that the approach of [8] encounters significant scalability challenges in generalizing to molecular systems of yet larger size.

5.2 Problem Formulation

We will work under the assumption of the so-called *Born-Oppenheimer approximation* [15], which treats the nuclei as fixed point charges. Indeed, the specification of a molecular geometry implicitly assumes such an approximation. When the positions of the nuclei are specified, the electronic structure problem can be restated as finding the ground eigenstate of the electronic Hamiltonian operator and consequently the ground state electronic energy becomes a parametric function of atomic positions. Here, the molecule's electronic Hamiltonian is commonly represented using the second-quantization formalism, with a chosen basis of atomic orbitals which describe the wave function of electrons in the molecule. In order to identify the Hamiltonian for a compound, we start by fetching the information required to build the target molecule object, then employ established solvers to compute the second-quantized Hamiltonian and store the result in a string format.

In the second-quantized formulation, the Hamiltonian is represented as a complex conjugate-symmetric (Hermitian) matrix H of side length 2^n where n represents the number of orbitals, and the basic problem is to determine the ground energy of H and a description of an associated eigenvector. Since storage and manipulation of such matrices is prohibitive, a common practice is to exploit the fact that H admits an efficient description in terms of superpositions of tensor products of 2×2 matrices drawn from the set $\{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}$, where

$$\sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad , \qquad \sigma_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad , \qquad \sigma_2 = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad , \qquad \sigma_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{5.1}$$

In fact, any Hermitian matrix $H \in \mathbb{C}^{2^n \times 2^n}$ admits a unique decomposition of the form

$$H = \sum_{\mathbf{p} \in \{0,1,2,3\}^n} \alpha_{\mathbf{p}} \, P_{\mathbf{p}} \tag{5.2}$$

where $P_{\mathbf{p}} := \sigma_{p_1} \otimes \cdots \otimes \sigma_{p_n}$ denotes a tensor product of Pauli matrices, and the real-valued coefficients entering the sum are determined by the formula $\alpha_{\mathbf{p}} = \frac{1}{2^n} \operatorname{tr}(H P_{\mathbf{p}})$. Since each matrix $P_{\mathbf{p}}$ is row-sparse with exactly one nonzero entry per row, the number of nonzero entries per row of H is bounded by the number of nonzero coefficients K entering the above sum, which could be as large as 4^n for typical matrices. The matrices of relevance to quantum chemistry are far from typical, however, and consequently enjoy a high level of row sparsity, which will be a crucial property in our subsequent algorithm development.

Beyond the assumption of row-sparsity, additional features of H can be determined

from the structure of the constituent Pauli strings $\mathbf{p}$, which in turn depend on the choice of encoding map from the chemical Hamiltonian to the qubit representation. Efficient qubit encoding maps are an active field of research, and the most common examples in usage are Jordan-Wigner [64] and Bravyi-Kitaev [19], which are equivalent isospectrally. In this work, we will adopt the Jordan-Wigner encoding because of its simplicity and the fact that it has shown success in prior work on VMC applied to ab-initio quantum chemistry [28, 8]. In the particular case of Jordan-Wigner transformation, we point out that many of the tensor factors of terms appearing in the sum (5.2) are the identity matrix and are moreover organized in a hierarchical structure.

Following the standard neural-network variational Monte Carlo procedure, we postulate a family of trial wavefunctions whose complex amplitudes relative to the standard basis are computed by the output of a neural network, parametrized by variational parameters $\theta \in \mathbb{R}^d$. Thus, given a function of the form

$$f : \{0,1\}^n \times \mathbb{R}^d \longrightarrow \mathbb{C} \tag{5.3}$$

which is differentiable in the second argument, we define an associated family of trial quantum states $|f_\theta\rangle$, which are differentiably parametrized by $\theta \in \mathbb{R}^d$, as follows

$$|f_\theta\rangle := \sum_{x \in \{0,1\}^n} f(x,\theta)|x\rangle \tag{5.4}$$

where $|x\rangle := |x_1\rangle \otimes \cdots \otimes |x_n\rangle$ is a shorthand to denote the standard basis vectors for $\mathbb{C}^{2^n}$. In this work, following [8], we assume that the function f is chosen in such a way that $|f_\theta\rangle$ is unit-normalized for all $\theta \in \mathbb{R}^d$, and that exact sampling from the probability distribution $\pi_\theta := |f(\cdot,\theta)|^2$ is computationally tractable. Using the unit vector (5.4) as a trial vector in the Rayleigh quotient for H, we obtain a differentiable objective function $\mathcal{L} : \mathbb{R}^d \longrightarrow \mathbb{R}$, which upper bounds the minimal eigenvalue $\lambda_{\min}(H)$ as a consequence of the Rayleigh-Ritz principle,

$$\mathcal{L}(\theta) := \frac{1}{2}\langle f_\theta|H|f_\theta\rangle \tag{5.5}$$

and whose value can be estimated at the Monte Carlo rate using the following estimator

$$\mathcal{L}(\theta) = \frac{1}{2}\mathbb{E}[l_\theta(x)] \ , \qquad l_\theta(x) := \frac{\langle x|H|f_\theta\rangle}{f(x,\theta)} \ , \qquad x \sim \pi_\theta \tag{5.6}$$

where l_θ is referred to as the local energy. Minimization of $\mathcal{L}$ is performed using stochastic gradient-based optimization techniques. In particular, the gradient of $\mathcal{L}$ can be estimated at

the Monte Carlo rate using,

$$\nabla\mathcal{L}(\theta) = \mathrm{Re}\,\mathbb{E}\big[\big(l_\theta(x)\mathbb{1} - b\big)\overline{\sigma_\theta}(x)\big] \quad , \qquad \sigma_\theta(x) := \frac{\nabla_\theta f(x,\theta)}{f(x,\theta)} \quad , \qquad x \sim \pi_\theta \tag{5.7}$$

where $b \in \mathbb{R}^{d\times d}$ is an arbitrary baseline matrix that can be set to $b = \mathbb{E}[l_\theta(x)]$.

Eq. (5.7) demonstrates that an integral component of the algorithm is the computation of the local energy, which is required both to perform stochastic gradient updates of the model and to estimate the value of the objective function. The computational complexity of computing the mapping $x \longmapsto l_\theta(x)$ for a minibatch of size B is evidently $O(BK)$ where recall that K denotes the number of terms in the Hamiltonian expansion (5.2). Despite the fact that the sparsity parameter satisfies $K \ll 4^n$, the computation of local energy still suffers from severe OOM issues in practice since 4^n can be extraordinarily large even for modestly sized molecules. For example, in the case of Methanol with $n = 28$ orbitals we have $4^n \approx 7.2 \times 10^{16}$ while $K = 52887$. This implies that for Methanol, given a modest batch size $B = 1024$, the local energy needs a forward pass of 54M samples of input size 28. This computational bottleneck is inevitable as large batch sizes are essential to ensure the precision of the Monte-Carlo approximation of the energy objective in Eq. (5.6), which directly influences the performance of the algorithm. Therefore, efficient sampling and local energy computation become the key issue for scaling the neural-network variational approach to large compounds.

5.3 Autoregressive Modeling of Molecular Quantum States

Motivated by the fact that solving high dimensional molecular quantum systems is still a difficult problem with existing variational techniques, we consider the base model from Zhao *et al.* [152], which was shown to be capable of solving quantum systems of very high dimensions. Adaptation of this work to chemistry problems involves generalizing from real-valued to complex-valued wave functions, as well as adjusting the sampling process to handle larger numbers of configurations and accommodate domain priors. In addition, following [8], we enforce constraints necessary to ensure that the generated samples correspond to physical electronic states.

Architecture. Since the ground state of the targeted problem is in general complex-valued, we consider the complex generalization of the model in [152] by splitting the output of the wave function into modulus and phase parts, learned by two sub-models separately as

follows

$$\begin{aligned}
\text{Modulus sub-model: } \textit{Input} &\xrightarrow{[B,N]} \texttt{MaskedFC1} \xrightarrow{[B,h]} \texttt{ReLU} \\
&\xrightarrow{[B,h]} \texttt{MaskedFC2} \xrightarrow{[B,N]} \texttt{Sigmoid} \xrightarrow{[B,N]} \textit{Output}, \\
\text{Phase sub-model: } \textit{Input} &\xrightarrow{[B,N-2]} \texttt{MaskedFC1} \xrightarrow{[B,h]} \texttt{ReLU} \\
&\xrightarrow{[B,h]} \texttt{MaskedFC2} \xrightarrow{[B,4]} \textit{Output}.
\end{aligned} \tag{5.8}$$

The modulus model predicts N conditional probabilities for each configuration, and the phase model predicts the phases for each of the four configurations that are identical in the first $N-2$ entries, which is a $N-2$ dimensional vector being fed as the input. Here B is the batch size, n is the number of dimensions, h is the hidden layer size and `MaskedFC` is the masked fully connected layer, which removes the connections in the computational path of MADE. The outputs of modulus sub-model are the conditional probabilities $\{\pi_i(x_i|x_{i-1},\ldots,x_1)\}_{i=1}^n$, which together define a joint probability function $\pi : \{0,1\}^n \longrightarrow [0,\infty)$ for input strings x. Normalization follows automatically from the autoregressive assumption,

$$\pi(x) = \prod_{i=1}^{n} \pi_i(x_i|x_{i-1},\ldots,x_1) \ . \tag{5.9}$$

On the other hand, we model the phase $\phi : \{0,1\}^n \longrightarrow [0,2\pi)$ of the wavefunction directly by feeding the input x into a two-layer MLP. It follows that the complex logarithm of the model output $f(x,\theta)$ can be written in a computationally tractable form as

$$f(x,\theta) = e^{i\phi(x)} \prod_{i=1}^{n} \sqrt{\pi_i(x_i|x_{i-1},...,x_1)}, \tag{5.10}$$

$$\log f(x,\theta) = i\phi(x) + \frac{1}{2}\sum_{i=1}^{n} \log \pi_i(x_i|x_{i-1},...,x_1) \tag{5.11}$$

where $\theta \in \mathbb{R}^d$ denotes the concatenation of parameters describing the neural networks ϕ and π.

Sampling. Standard autoregressive sampling techniques such as NADE [75] have a sequential nature that updates a batch of randomly initialized states entry by entry following a pre-fixed order. In practice, this approach is highly inefficient as the sampled batch usually contains repeated samples. Instead, we keep track of a sample buffer throughout the sampling process, associated with a counter storing the number of occurrences for each sample

in the buffer. We start with a buffer containing only one random initialized sample. At each of the n iterations, we first double the size of the buffer by alternating ± 1 value for existing samples at a fixed entry. Then, we update the counter for all samples in the buffer through Bernoulli sampling with probabilities computed by forward pass. Finally, we eliminate the samples in the batch that have the lowest numbers of occurrences in the counter, to avoid the exponential growth of the buffer size.

In addition, as a consequence of the Jordan-Wigner encoding, many of the tensor factors appearing with high qubit index are given by the identity factor, which leads to the expectation that high index qubits are comparatively less correlated compared to those with at low indices. Motivated by this observation, in an effort to ease the training of the model we performed autoregressive sampling in a reversed order beginning with the nth qubit.

Constraints. Recall that the unconstrained model assigns nonzero probability mass to all 2^n possible bit-strings representing possible states of a multi-electron system. However, in quantum chemistry problems we considered, only $n_e \leq n$ out of the n single-electron spin-orbitals are occupied with electrons, and the net charge C of the molecular system determines the number of unpaired electrons. It follows that the numbers of electrons with up-spins and down-spins $n_\uparrow, n_\downarrow$ respectively should satisfy

$$n_\uparrow + n_\downarrow = n_e, \quad n_\uparrow - n_\downarrow = C. \tag{5.12}$$

This corresponds to applying Hamming weight constraints to the bit-strings, which effectively reduces the total number of candidate samples from 2^n to $\binom{n/2}{n_\uparrow}\binom{n/2}{n_\downarrow}$, which significantly reduces the complexity of the problem. We adopted techniques from similar work [60, 8] to enforce the constraints. Define the hamming weight constraint of the configuration $x \in \{0,1\}^n$ to be

$$\sum_{i=1}^{n/2} x_{2i-1} = n_\uparrow, \quad \sum_{i=1}^{n/2} x_{2i} = n_\downarrow, \tag{5.13}$$

where the even and odd indices correspond to the up-spin and down-spin electrons in the orbitals. The idea to enforce the constraint in the autoregressive sampling process is to assign nonzero probabilities only to samples that satisfy Eq. (5.13), for which the sufficient and necessary condition is that the kth entry x_k must satisfy

$$n_\uparrow - (n/2 - \lceil k/2 \rceil) \leq \sum_{i=1}^{\lceil k/2 \rceil} x_{2i-1} \leq n_\uparrow, \quad n_\downarrow - (n/2 - \lceil k/2 \rceil) \leq \sum_{i=1}^{\lceil k/2 \rceil} x_{2i} \leq n_\downarrow, \tag{5.14}$$

for all $k \in 1, 2, ..., N$. Condition (5.14) can be enforced at every iteration k during the sampling process by introducing the following modified probability distribution,

$$\widehat{\pi}(x_k = 1|x_{k-1}, ..., x_1) = \begin{cases} 1, & \text{if } \sum_{i=1}^{\lceil k/2 \rceil} x_{2i-1} < n_\uparrow - (\frac{n}{2} - \lceil k/2 \rceil) \\ 1, & \text{if } \sum_{i=1}^{\lceil k/2 \rceil} x_{2i} < n_\downarrow - (\frac{n}{2} - \lceil k/2 \rceil) \\ 0, & \text{if } \sum_{i=1}^{\lceil k/2 \rceil} x_{2i-1} \geq n_\uparrow \\ 0, & \text{if } \sum_{i=1}^{\lceil k/2 \rceil} x_{2i} \geq n_\downarrow \\ \pi(x_k = 1|x_{k-1}, ..., x_1), & \textit{o.w.}, \end{cases} \tag{5.15}$$

which has the property that for any x violating the constraints, we have

$$\widehat{\pi}(x_k = x|x_{k-1}, ..., x_1) = 0, \quad \widehat{\pi}(x_k = 1 - x|x_{k-1}, ..., x_1) = 1. \tag{5.16}$$

The modified probabilities still normalize to one by design.

5.4 Parallelization

We apply parallelization to our algorithm from two perspectives. We parallelize the training by trunking the input batch across the GPUs, where the model parameters are replicated on each GPU, which handles a portion of the full batch. During the backward pass, gradients from each node are averaged. Locally within each process, we tensorize the entire computational pipeline so that the computation of local energies is fully GPU-supported.

5.4.1 Tensorized Computation of Local Energy

The form of the local energy has the property that it can easily be parallelized, which becomes increasingly important with increasing molecular system size since the Hamiltonian can potentially involve a large number of terms. We implemented an efficient tensor representation of the second quantized spin Hamiltonian generated from chemical data, which capitalizes on the fact that the matrix corresponding to an arbitrary product of Pauli operators $P_{\mathbf{p}}$ is extremely sparse In particular, for each row index $x \in \{0, 1\}^n$ there is exactly one column index $x^{\mathbf{p}} \in \{0, 1\}^n$, for which the corresponding matrix entry $\langle x|P_{\mathbf{p}}|x^{\mathbf{p}}\rangle$ is nonzero[1]. If we denote the subset of Pauli strings with nonzero coefficient by

[1]The formula for the associated matrix entry in terms of x is given in [28].

$S = \{\mathbf{p} \in \{0,1,2,3\}^n : \alpha_{\mathbf{p}} \neq 0\}$, then the local energy simplifies to

$$l_\theta(x) = \sum_{\mathbf{p}\in S} \alpha_{\mathbf{p}} \langle x|P_{\mathbf{p}}|x^{\mathbf{p}}\rangle \frac{f(x^{\mathbf{p}},\theta)}{f(x,\theta)} \tag{5.17}$$

The goal is now to efficiently compute Eq. (5.17), in which the number of summands $K = |S|$ is very large. The idea, depicted in Fig. 5.1, is to extract the key information required to perform the computation from the molecular Hamiltonian and store the information as tensors that directly support GPU computation. To this end, we constructed a string parser that computes an Operator Matrix along with coefficients. For Pauli string, we track the indices of Pauli $\sigma_1, \sigma_2, \sigma_3$ operators as tensors, which are later utilized to compute $x^{\mathbf{p}}$ and the corresponding matrix element $\langle x|P_{\mathbf{p}}|x^{\mathbf{p}}\rangle$.

In practice, $x^{\mathbf{p}}$ is computed by flipping the bits of x based corresponding to the locations of σ_1 and σ_2 in $P_{\mathbf{p}}$. To improve the efficiency, we collect the indices of σ_1, σ_2 operators prior to the training and only keep a buffer of unique flippings and their corresponding number of occurrences. This approach can effectively reduce the input size to the forward pass by removing the repeated samples, which is particularly helpful for large-sized problems. The matrix element $\langle x|P_{\mathbf{p}}|x^{\mathbf{p}}\rangle$ admits the formula

$$\langle x|P_{\mathbf{p}}|x^{\mathbf{p}}\rangle = x|\sigma_{p_1} \otimes \cdots \otimes \sigma_{p_n}|x^{\mathbf{p}} = (-i)^r \prod_{k:p_k\in\{2,3\}} (-1)^{x_k} \tag{5.18}$$

where r is the number of occurrences for the σ_2 operator, *i.e.*, $r = \sum_{k=1}^n \mathbb{1}_{\{p_k=2\}}$. Similar as before, the indices of σ_2, σ_3 operators are collected prior to the training, and the product can be calculated by the Hadamard product between the indices and x, followed by production of all entries. Note that all computations in this subsection can be performed in parallel for all terms with GPU.

5.4.2 Parallelization

In applications such as quantum chemistry, the Hamiltonian of even the smallest molecules contain thousands of terms, which lead to severe OOM issues for the existing VMC platforms. Our proposed pipeline tensorizes the information in the molecular Hamiltonian to maximize memory efficiency. In addition, the computation of local energy is conducted term-wise with no interaction between the terms, which motivates embarrassingly parallel algorithms for Hamiltonians consisting of a large number of terms. We make a step toward addressing the bottleneck by applying our sampling parallelization strategy to this problem, where we use identical copies of the model across the computing units to generate only a few samples

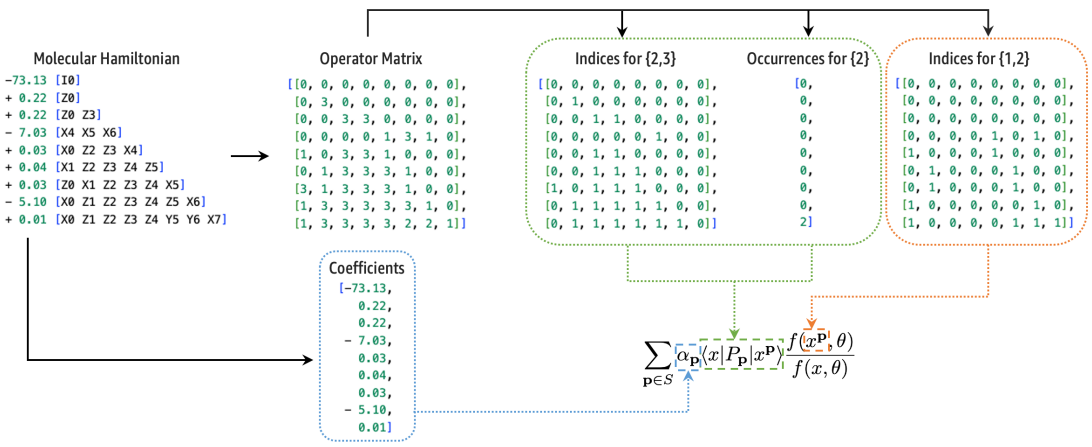


Figure 5.1: Local energy computational paradigm. The hamiltonian is parsed into an operator matrix, which gives indices and occurrences involved in the computation of Eq. 5.18, in the form of matrices. This formulation provides significant computational speed-up for molecular Hamiltonian with a huge number of terms.

per unit and combine the independent samples from all these units to construct an accurate expectation estimate. In addition, we remove the replicated configurations as described in the previous section before the forward pass locally for each GPU to save more memory.

In more detail, recall that the energy expectation is approximated as the following double sum

$$\frac{f_\theta|H|f_\theta}{f_\theta|f_\theta} \approx \frac{1}{B}\sum_{i=1}^{B}\sum_{\mathbf{p}\in S} \alpha_{\mathbf{p}} \langle x_i|P_{\mathbf{p}}|x_i^{\mathbf{p}}\rangle \frac{f(x_i^{\mathbf{p}},\theta)}{f(x_i,\theta)} \quad . \tag{5.19}$$

We distribute the BK summands in the above sum across L GPUs in order to perform thhe forward and backward passes of the model with a mini-batch size of BK/L. Locally, each GPU has access to to the necessary ingredients required to compute the corresponding partial sums of size BK/L. We compute local gradients with forward and backward passes within each GPU and update the model parameters with the averages of local gradients obtained by averging over BK/L elements. In addition, we used gradient accumulation [78] splitting the batch into several mini-batches before a single update to avoid potential OOM issues for molecules with larger sizes.

5.5 Experiments

We now investigate the performance of our algorithm and the running time efficiency of the proposed parallelization paradigm. We first demonstrate our main results by comparing our algorithm with Hartree-Fock (HF) and CC with up to double excitations (CCSD) baselines,

Name	MF	n	$N_\uparrow + N_\downarrow$	K	HF Energy	CCSD	Ours	FCI
Hydrogen	H2	4	1+1=2	15	-1.06610864	**-1.101150**	**-1.101150**	-1.101150
Lithium Hydride	LiH	12	2+2=4	631	-7.76736213	-7.784455	**-7.784460**	-7.784460
Water	H2O	14	5+5=10	1390	-74.9644475	-75.015409	**-75.015511**	-75.015530
Methylene	CH2	14	5+3=8	2058	-37.4846329	-37.504411	**-37.504419**	-37.504435
Beryllium Hydride	BeH2	14	3+3=6	2074	-14.4432411	-14.472713	**-14.472922**	-14.472947
Ammonia	NH3	16	5+5=10	4929	-55.4547926	-55.520931	**-55.521037**	-55.521150
Methane	CH4	18	5+5=10	8480	-39.7265817	-39.806022	**-39.806170**	-39.806259
Diatomic Carbon	C2	20	6+6=12	2239	-74.2483215	-74.484727	**-74.486037**	-74.496388
Fluorine	F2	20	9+9=18	2951	-195.638041	**-195.661086**	-195.661067	-195.66108
Nitrogen	N2	20	7+7=14	2239	-107.498967	-107.656080	**-107.656763**	-107.660206
Oxygen	O2	20	9+7=16	2879	-147.631948	-147.747738	-147.749953	-147.750235
Lithium Fluoride	LiF	20	6+6=12	5849	-105.113709	-105.159235	**-105.165270**	-105.166172
Hydrochloric Acid	HCl	20	9+9=18	5851	-455.135968	**-455.156189**	**-455.156189**	-455.156189
Hydrogen Sulfide	H2S	22	9+9=18	9558	-394.311379	-394.354556	**-394.354592**	-394.354623
Formaldehyde	CH2O	24	8+8=16	20397	-112.354197	-112.498567	**-112.500944**	-112.501253
Phosphine	PH3	24	9+9=18	24369	-338.634114	-338.698165	**-338.698186**	-338.698400
Lithium Chloride	LiCL	28	10+10=20	24255	-460.827258	-460.847580	**-460.848109**	-460.849618
Methanol	CH4O	28	9+9=18	52887	-113.547027	**-113.665485**	**-113.665485**	-113.666485
Lithium Oxide	Li2O	30	7+7=14	20558	-87.7955672	-87.885514	**-87.885637**	-
Ethylene Oxide	C2H4O	38	12+12=24	137218	-150.927608	-151.120474	**-151.120486**	-
Propene	C3H6	42	12+12=24	161620	-115.657941	-115.885123	**-115.886571**	-
Acetic Acid	C2H4O2	48	16+16=32	461313	-224.805400	**-225.050896**	-225.0429767	-
Sulfuric Acid	H2O4S	62	25+25=50	1235816	-689.262656	-689.498410	**-689.505237**	-
Sodium Carbonate	CNa2O3	76	26+26=52	1625991	-575.016102	-575.299810	**-575.299820**	-

Table 5.1: Best molecular ground-state energies obtained by different methods as described in the main text over five trials. Molecules have been sorted according to the number of qubits used in the Jordan-Wigner representation. In addition, the numbers $(N_\uparrow, N_\downarrow)$ up- and down-spin electrons and the number K of terms in the Hamiltonian are reported. Our method exhibits superior performance in comparison with classical approximate methods such as Hartree-Fock and CCSD and come close to the FCI ground truth, which is only available up to molecules of size 28 qubits.

where our performance is either on par or superior to the classical approximate methods over a wide list of chemical molecules. Our performance is also close to the ground truth FCI energies up to molecular systems with 28 qubits, where the results for larger molecules become increasingly hard to obtain. We then perform ablation studies to examine our algorithm over various aspects. First, we show that increasing batch size improves the performance, at the cost of increased algorithmic complexity. On the other hand, our parallelization strategy can effectively reduce the running time, achieving near-optimal weak scaling. Our proposed sampling trick also improves the performance of our model by a noticeable margin. At last, we show that our model architecture exhibits superior running time efficiency compared against RBM [28] and NADE [8].

5.5.1 Experimental Set-ups

Given a target molecule ID, we fetch its corresponding *PubChem Compound Identifier (CID)* from the official PubChem website [139]. CID is recognized by `PubChemPy`, a software that provides a way to interact with *PubChem* in Python, allowing depiction and retrieval of chemical properties, such as the geometry of atoms, number of unpaired electrons, total charge, *etc*. The mapping from second-quantized Hamiltonian to interacting spinning model is done by transforming fermion operators into qubit operators with Jordan-Wigner [64] using `OpenFermion-Psi4` [85]. Note that these solvers can also be used to estimate the ground state energies including Hartree-Fock, CC with up to double excitations (CCSD), and FCI. The whole data processing pipeline is automatic without further human interference.

We train the model over 10K iterations with Adam optimizer [68] by default at a learning rate of 1×10^{-3} with standard decay rates for the first- and second-moment estimates of $\beta_1 = 0.9$ and $\beta_2 = 0.99$, respectively; no learning rate scheduler is applied. The batch size for the number of unique samples is fixed to be 1024 throughout our experiments unless specified otherwise. For scalability experiments, each GPU is distributed with a constant mini-batch size mB, and the effective batch size is $mB\times L$, where L is the total number of GPUs available. All experiments in this work use a single set of hyperparameters and identical training procedures, and the results are the best energies obtained across exactly 5 seeds. Throughout the experiments, the timing benchmarks are performed on Tesla V100-SXM2-16GB and Intel(R) Xeon(R) CPU @ 2.30GHz processors on Google Cloud Platform (GCP).

5.5.2 Performance Benchmark

We report the performance of our implementation over a wide variety of molecules in Table 5.1. The molecules are sorted according to the number of qubits in their Jordan-Wigner representation. We consider Hartree-Fock (HF) and CCSD energies as classical baseline approximate methods to compare against. The FCI ground truth energy is also provided for smaller molecules for reference.

Our model exhibits consistently strong performance on all molecules considered. In particular, the computed energies match the ground-truth FCI result closely on all molecules with up to 20 electrons and 28 spin-orbitals and out-perform other approximate methods on the majority of the molecules we considered. Notice that as the size of the molecule increases, the number of terms in the electronic Hamiltonian formulation also grows quickly, which leads to severe computational issues such as the running time and memory consumption. As a result, our proposed algorithm is capable of scaling up to system size with 76 qubits of

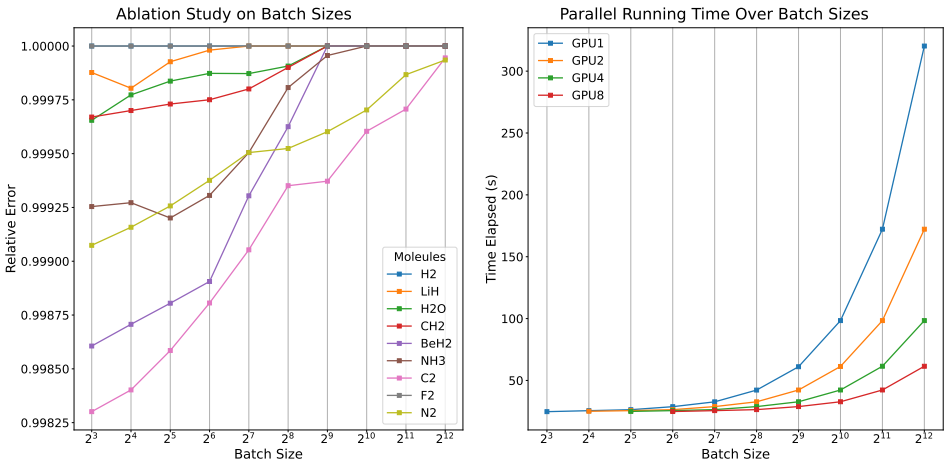


Figure 5.2: Left: The performance of our algorithm with increasing batch size for different molecules. The Relative Performance score is obtained by normalizing the estimated energy with respec to the corresponding FCI ground truth. Better performance is obtained by training with larger batch sizes, and the improvements become substantial for larger molecules. Right: Demonstration of near-optimal weak scaling obtained by running the algorithm on the C2 molecule for 10^3 iterations with different batch sizes up to 4096 and reporting the time elapsed in seconds. Upon distributing the batch over multiple GPUs, the running time is significantly reduced. In addition, the running times for the training with a fixed batch size per GPU are close across different settings.

which the electronic Hamiltonian is consisted of 1.6 million terms, and achieves state-of-the-art performance.

5.5.3 Ablation Studies

In this section, we perform ablation studies to test the effectiveness of the ingredients in our contribution. We start by validating the fact that increasing batch size is capable of improving the performance for larger scale problems, which justifies the motivation of our parallelization scheme that enables large-batch training for large molecules. We also examine our proposed reverse sampling trick by comparing the performance with different sampling orders. Finally, we tried different model architectures under the same training framework; our model achieves the overall best running time efficiency in comparison with RBM [28] and NADE [8].

Parallelization. As discussed in Section 5.2, the total number of input samples for the forward pass required to compute the local energy scales with the number of terms K in the Hamiltonian and with the batch size B, which leads to a heavy computational bottleneck as both of these factors increase. However, sufficiently large batch size is essential to guarantee the performance of the algorithm for molecules of larger sizes. This claim is validated in left panel of Figure 6.2, where we train our model with batches of varying size for various

Molecule	H2	H2O	NH3	C2	N2	O2	HCL
Energy	Ablation study on the sampling order						
Forward	**-1.101150**	-75.015449	-55.515394	-74.4849249	-107.634908	-147.723681	-454.927860
Reverse	**-1.101150**	**-75.015511**	**-55.521037**	**-74.4860377**	**-107.656763**	**-147.749953**	**-455.156189**
Random	**-1.101150**	-75.014553	-55.519741	-74.4851979	-107.606417	-147.732876	-455.012498
Running time (s)	Running time for 30K iterations						
NADE	303.76	1087.52	4474.94	4574.30	2959.28	2821.60	1593.21
MADE	**282.64**	**838.60**	**3188.31**	**2922.25**	**2295.90**	**2122.21**	**1112.70**
Hitting time (s)	Hitting time to the CCSD performance						
NADE	117.28	**352.78**	2007.14	1754.10	1029.64	824.37	489.88
MADE	**100.14**	364.32	**782.46**	**827.41**	**986.27**	**648.27**	**186.38**

Table 5.2: Ablation study on reverse sampling and time efficiency tests over different architectures.

molecules. Since the ground state energies for different molecules differ from each other, we report the performance relative the the FCI ground truth.

To examine the effectiveness of our parallelization scheme, in the right panel of Figure 6.2 we illustrate the running time as a function of batch size for C2 molecule using 10^3 iterations. We observe that the running time scales inversely with respect to the number of GPUs. In particular, doubling the number of GPUs roughly corresponds to half the time usage. We conducted additional experiments by saturating the memory on each GPU and observe that the running time for different numbers of GPUs remains constant, indicating that our approach achieves near-optimal weak scaling.

Reverse Order Sampling. We proposed to perform autoregressive sampling from a reversed order to improve the training. To examine the effectiveness of this approach as well as the impact of the sampling order on the quantum chemistry problems in general, we perform ablation studies on the sampling order. In Table 5.2, we employ three different sampling orders: forward sampling from 1 to n, reverse sampling from n to 1, and random sampling by any pre-determined order between 1 to n. The results illustrate that reverse sampling indeed improves the performance effectively as it achieves the best results consistently across the list of molecules.

Model Architecture. Our model architecture offers significant parllelization advantages compared to existing architectures based on RBM and NADE. Despite the sequential nature of both MCMC and autoregreesive sampling, the latter can be executed on GPUs in a straightforward fashion and moreover exhibits superior running time efficiency. In addition, the distribution of the MCMC samples only converges to the distribution of interest asymptotically, whereas autoregressive sampling yields exact samples under a known number of iterations. NADE [75] requires n forward passes through the network to

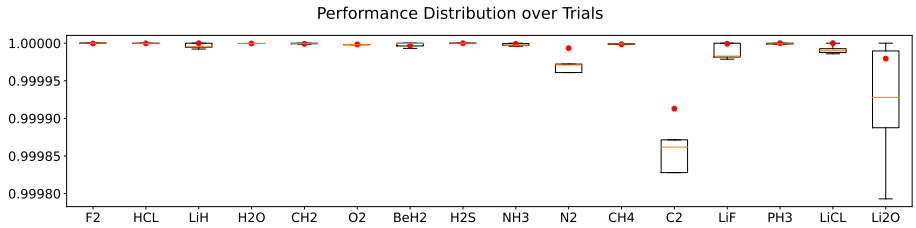


Figure 5.3: We report our performance on different molecules for 5 trials in the form of a box plot. For illustration purposes, we divide the results for all molecules by the corresponding FCI ground truths, so that the reported value is normalized with a maximum value of 1. In addition, we directly cite the numbers of the state-of-the-art and mark them in the form of red dots in the figure for comparison purposes. We notice that for certain molecules, the performance of our method fluctuates due to different random seeds. Nevertheless, competitive results can be obtained after sufficient number of trials.

evaluate the probability, with n submodules for each entry. The main disadvantage of NADE is its sequential nature in its forward pass, which contributes to running time as the input dimension grows. In addition, computation with NADE is slower in practice compared to MADE even in low dimension, especially for model of high depth, due to its multi-module architectural design. In Table 5.2, we directly compare the time taken for NADE and MADE to run for 3×10^4 iterations. In addition, we measure the time taken for each architecture to reach the performance of CCSD for each molecule. We did not include the results for RBM for two reasons. First, the actual running times of RBM for 30K iterations are exceedingly large, *e.g.*, about four hours for H2 molecule, therefore the direct comparison results against the other two architectures have no practical interest. Second, the performance of RBM is usually inferior to CCSD, so the hitting time is not available. On the other hand, MADE exihibits a significant computational advantage compared to NADE in both measures. We also directly cite the numbers from the Table 1 in [8] and compare the results with ours in Figure 5.3 for a sanity check.

5.6 Conclusion

We proposed a scalable parallelization strategy to improve the VMC algorithm in the application of ab-initio quantum chemistry. Our local energy parallelism enables the optimization for Hamiltonians of more complex molecules and our autoregressive sampling techniques out-perform the CCSD baseline and exhibits an advantage against other neural-network based algorithms in terms of running time efficiency and scalability. We further improve the performance of our model through the sampling order of the state entries to match the entanglement hierarchy among the molecule qubits. Our algorithm effectively

works for molecules up to 76 qubits with millions of terms in its electric Hamiltonian.

Chapter 6

Application: Financial Derivative Pricing

The field of stochastic variational algorithms has undergone dramatic recent developments based on two relatively recent, albeit independent, developments: (*i*) the availability of near-term quantum computers [102] and (*ii*) the existence of scalable stochastic algorithms for training deep neural networks. The former development has motivated a new research direction called variational quantum algorithms (VQAs) [27], in which the stochastic variational character of the algorithms render them suitable for noisy intermediate-scale quantum computers [102]. The latter holds promise to accelerate a host of scientific computing problems including general-purpose solvers of partial differential equations (PDEs) using physics-informed neural-networks (PINNs) [104] and has already made substantial headway in solving the time-(in)dependent Schrödinger equation in high dimensions using variational quantum Monte Carlo (VMC) with neural-network quantum states [24]. It is noteworthy that there exist many shared parallels between the fields of VQAs, VMC [127, 129] and PINNs. In particular, both VMC and VQAs hinge on the concept of adaptive stochastic estimation of the time-independent and time-dependent variational principles originally due to Rayleigh-Ritz and McLachlan [87], respectively. VQAs and VMC differ essentially in the choice of parametrized quantum state and the associated adaptive sampling strategy used to estimate quantum expectation values. PINNs can be viewed as an alternative approach to McLachlan's variational principle. They gained significant traction outside the quantum physics literature and have been advocated for solving general PDEs that appear in other areas of science and engineering. This approach involves solving a non-convex optimization problem for the PDE residual in order to determine the space-time development of the state variable. A large number of variants of this proposal have been subsequently put forward targeting PDEs either in many spatial dimensions (e.g., [123, 138]) or parametric PDEs in low dimensions (e.g., [77]). Deep backward stochastic differential equation methods [56, 57, 12, 62] are another class of deep learning approaches that have been applied to parabolic PDEs that arise in mathematical finance.

In this section, we introduce a generalization of McLachlan's variational principle applicable to a wide variety of time-dependent PDEs and propose a variational quantum Monte Carlo (VMC) stochastic approximate solution method utilizing autoregressive neural-network quantum states. A closely related algorithm has been recently introduced in the VQA literature [38, 4], which was motivated by the problem of solving linear PDEs using digital quantum computers. Ref. [4] argued for an approximate solution concept, with respect to which exponential quantum speedup could be achieved for state evolution, which however is overwhelmed by auxiliary costs of state preparation and extraction of properties of the state (see [4, Section III.C] for details). These complications of read-out and state preparation are not relevant to the VMC-based solver, however, since the VMC computing model permits efficient queries to arbitrary probability amplitudes. Interestingly, our generalization of McLachlan's variational principle is closely related to the neural Galerkin method recently put forward in [20], which also introduced a different neural-network-based stochastic solution method. One small difference compared to [20] is that we choose to work on a predefined mesh, which provides a simple method to implement non-trivial boundary conditions necessary for financial applications. The use of a mesh is not mandatory, however, and this section paves the way toward a mesh-free generalization. Indeed, in the final stages of preparation of this article, Ref. [107] appeared, which proposes a mesh-free flow-based solution of probabilistic PDEs such as the Fokker-Planck equation. Unlike [107], however, our approach does not require that the state variable of the underlying PDE corresponds to a probability density.

The speedup obtainable by the approach advocated here has practical applications in overcoming the curse-of-dimensionality in high-dimensional PDEs, particularly in situations where fine-grained information about the state variable is required, such as gradients with respect to the independent variables. One such situation is quantum many-body physics, where kinetic energy observables depend on second-order spatial derivatives. Another example is the pricing and hedging of contingent claims in multi-asset financial markets. The approach is applicable to general time-dependent PDEs expressible in first-order form, although we only consider the inhomogeneous linear case in the numerical experiments. Specifically, we chose to focus on the multi-asset Black-Scholes PDE for pricing European contingent claims because of its well-known relation with the time-dependent Schrödinger equation (TDSE) as well as its importance in computational finance. This work opens the door to free-boundary problems necessary for pricing American contingent claims.

The organization of the section is as follows: In section 6.1 we generalize the McLachlan variational principle to general time-dependent PDEs in a model-agnostic manner, which is applicable in the purely classical or quantum setting. In section 6.2 we describe the modeling

assumptions involved in the passage from a time-dependent PDE to an neural quantum state-based solution of McLachlan's variational principle. The remainder of the section is dedicated to numerical experiments, focusing on the problem of financial derivative pricing, which suffers from a curse-of-dimensionality. Section 6.3 in particular provides numerical confirmation in the case of correlated diffusions, and section 6.4 describes the application of these results to option pricing in the Black-Scholes pricing framework.

6.1 Theory

6.1.1 Generalities of McLachlan's variational principle

McLachlan's variational principle [87] is an example of a time-dependent variational principle (TDVP) that approximates the solution of the TDSE by evolution within a space of parametrized trial functions. TDVPs for the TDSE have been devised for tensor network states [52], neural-network quantum states (NQS) [24] and parametrized quantum circuits [127, 7]. Ref. [129] studied TDVPs through the lens of information geometry providing a unified perspective applicable to variational quantum algorithms (VQAs) and variational quantum Monte Carlo with normalized neural-network quantum states. In the following section, we further generalize TDVPs to include general time-dependent PDEs and establish additional connections with the VQA and numerical analysis literature. Given a time-dependent PDE for the state variable $u(t,x) \in \mathbb{C}$ with $(t,x) \in [0,T] \times \Omega$, together with a choice of parametrized functions of the spatial variable $\{u_\theta : \Omega \longrightarrow \mathbb{C} \,|\, \theta \in \mathbb{R}^p\}$, the output of a TDVP is parametrized curve $\gamma : [0,T] \longrightarrow \mathbb{R}^p$ in the space of variational parameters such that $u_{\gamma(t)}$ optimally describes $u(t,\cdot)$ in some distance metric $\mathrm{dist}(\cdot,\cdot)$ for all $t \in [0,T]$. Consider the initial value problem for a general time-dependent PDE of the form,

$$\partial_t u(t,x) = \mathcal{F}(t,x,u) \tag{6.1}$$

$$u(0,x) = u_0 \in L^2(\Omega;\mathbb{C}) \ , \tag{6.2}$$

with prescribed boundary conditions on $\partial\Omega$. In order to simplify exposition, in this section we avoid complications of boundary conditions by assuming either $\Omega = \mathbb{R}^d$ with suitable decay at infinity or $\Omega = \mathbb{T}^d$. In practice we replace the spatial domain by a mesh $\widehat{\Omega} \subset \Omega$, thereby approximating square-integrable functions by square summable vectors. The imposition of boundary conditions on the spatial mesh is then achieved via the use of source functions. Let Φ_s^t denote the time evolution map for state variable such that $\Phi_u^t \circ \Phi_s^u = \Phi_s^t$

and in particular $u(t,\cdot) = \Phi_0^t(u_0)$. Given an initial parameter vector $\theta_0 \in \mathbb{R}^p$ and a step size $\delta t > 0$, define a sequence of parameter vectors $(\theta_k)_{k\in\mathbb{N}}$ by the following iteration

$$\theta_{k+1} := \operatorname*{arg\,min}_{\theta\in\mathbb{R}^p} \left[\operatorname{dist}\left(\Phi_{k\delta t}^{(k+1)\delta t}(u_{\theta_k}), u_\theta\right)\right] . \tag{6.3}$$

In quantum physics applications, a suitable distance metric is the Fubini-Study metric, as previously argued both in VMC [24, 129] and in VQA [127, 7] literature. In this work we choose $\operatorname{dist}(\cdot,\cdot)$ to be the Euclidean norm. Rather than solving the discrete-time dynamical system (6.3) directly, we consider the limit of infinitesimal step size $\delta t \longrightarrow 0$ in which it reduces to the following system of ordinary differential equations (ODEs) with initial condition $\gamma(0) = \theta_0$:

$$M(\gamma(t))\,\gamma'(t) = V(t,\gamma(t)) \tag{6.4}$$

where

$$M_{ij}(\theta) := \operatorname{Re}\left[\left\langle \frac{\partial u_\theta}{\partial\theta_i}\middle|\frac{\partial u_\theta}{\partial\theta_j}\right\rangle\right] \quad , \quad V_i(t,\theta) := \operatorname{Re}\left[\left\langle \frac{\partial u_\theta}{\partial\theta_i}\middle|\mathcal{F}(t,u_\theta)\right\rangle\right] \tag{6.5}$$

and where $\langle\cdot|\cdot\rangle$, $\|\cdot\|_2$ denote the standard inner product and the induced norm for $L^2(\Omega,\mathbb{C})$, respectively. Although the matrix M_{ij} is necessarily positive semi-definite, it may be degenerate, reflecting the possibility of multiple minima in the optimization problem (6.3). Thus, regularization techniques are generally required in order to obtain a well-posed system of ODEs.

6.1.2 Derivation of evolution equations

Assume that the time evolution map and the variational trial function admit Taylor expansions of the form

$$\Phi_t^{t+\delta t}\big(u\big) = u + \mathcal{F}\big(t,u\big)\delta t + \mathcal{O}(\delta t^2) \ , \tag{6.6}$$

$$u_{\theta+\delta\theta} = u_\theta + \sum_{i=1}^{p} \frac{\partial u_\theta}{\partial\theta_i}\delta\theta_i + \mathcal{O}(\delta\theta^2) \ . \tag{6.7}$$

Then

$$\begin{aligned}
&\left\|\Phi_t^{t+\delta t}(u_\theta) - u_{\theta+\delta\theta}\right\|_2^2 \qquad (6.8)\\
&= \sum_{i,j=1}^{p} \left\langle \frac{\partial u_\theta}{\partial \theta_i} \middle| \frac{\partial u_\theta}{\partial \theta_j} \right\rangle \delta\theta_i \delta\theta_j - \sum_{i=1}^{p} \left[\left\langle \mathcal{F}(t,u_\theta) \middle| \frac{\partial u_\theta}{\partial \theta_i} \right\rangle + \left\langle \frac{\partial u_\theta}{\partial \theta_i} \middle| \mathcal{F}(t,u_\theta) \right\rangle \right] \delta t\, \delta\theta_i + \cdots \\
&= \sum_{i,j=1}^{p} \frac{1}{2} \left[\left\langle \frac{\partial u_\theta}{\partial \theta_i} \middle| \frac{\partial u_\theta}{\partial \theta_j} \right\rangle + \left\langle \frac{\partial u_\theta}{\partial \theta_j} \middle| \frac{\partial u_\theta}{\partial \theta_i} \right\rangle \right] \delta\theta_i \delta\theta_j \\
&\qquad\qquad - \sum_{i=1}^{p} \left[\left\langle \mathcal{F}(t,u_\theta) \middle| \frac{\partial u_\theta}{\partial \theta_i} \right\rangle + \left\langle \frac{\partial u_\theta}{\partial \theta_i} \middle| \mathcal{F}(t,u_\theta) \right\rangle \right] \delta t\, \delta\theta_i + \cdots \\
&= \sum_{i,j=1}^{p} M_{ij}(\theta) \delta\theta_i \delta\theta_j - 2\delta t \sum_{i=1}^{p} \delta\theta_i V_i(t,\theta) + \cdots \qquad (6.9)
\end{aligned}$$

where in the last line we used the conjugate-symmetry of $\langle \cdot | \cdot \rangle$ and we have neglected $\delta\theta$-independent terms and terms higher than quadratic order in the multi-variable Taylor expansion in $\delta\theta$ and δt. The first-order optimality condition $0 = \frac{\partial}{\partial \delta\theta_i} \left\|\Phi_t^{t+\delta t}(u_\theta) - u_{\theta+\delta\theta}\right\|_2^2$, gives, at lowest order in $\delta\theta$ and δt,

$$0 = 2 \sum_{j=1}^{p} M_{ij}(\theta) \delta\theta_j - 2V_i(t,\theta)\delta t + \cdots \qquad (6.10)$$

and thus taking the limit $\delta t \longrightarrow 0$ gives the result.

6.1.3 Matrix representation of $\mathcal{L}$

In d spatial dimensions and multi-index $\mathbf{i} = \{i_1, i_2, \ldots, i_d\}$, let $\mathbf{i} \pm \mathbf{e}_k = \{i_1, i_2, \ldots, i_k \pm 1, \ldots, i_d\}$ and $\mathbf{i} \pm \mathbf{e}_k \pm \mathbf{e}_{k'} = \{i_1, i_2, \ldots, i_k \pm 1, \ldots, i_{k'} \pm 1, \ldots, i_d\}$. Notice we do not allow $+1$ if $i_k = \frac{n}{d}$ or -1 if $i_k = 1$. Then the elements of the matrix is given by:

$$[\widehat{\mathcal{L}}]_{\mathbf{i},\mathbf{j}} = \begin{cases} \frac{-d}{\Delta^2}, & \mathbf{j} = \mathbf{i}, \\ \frac{1}{2\Delta^2}, & \mathbf{j} = \mathbf{i} \pm \mathbf{e}_k, \\ \frac{D}{4\Delta^2}, & \mathbf{j} = \mathbf{i} \pm \mathbf{e}_k \pm \mathbf{e}_{k'}, \\ \frac{-D}{4\Delta^2}, & \mathbf{j} = \mathbf{i} \pm \mathbf{e}_k \mp \mathbf{e}_{k'}, \\ 0, & \text{otherwise.} \end{cases} \qquad (6.11)$$

6.1.4 Analogy with finite element approximations

Before discussing our autoregressive neural-network quantum state implementation, it is useful to orient within scientific computing literature by showing that the formalism shares close parallels with the classic finite element method applied to the linear inhomogeneous case $\mathcal{F}(t,x,u) = \mathcal{L}u(t,x) + f(t,x)$. Given a set of real basis functions $\{\varphi_i\}_{i=1}^m$, define the variational family consisting of a weighted superposition,

$$u_\theta(x) = \sum_{i=1}^{m} \theta_i \varphi_i(x) \ , \tag{6.12}$$

where $\theta \in \mathbb{R}^m$ is assumed here. Substituting the above into (6.4) one finds the following ODE determining the dynamics of the weights,

$$M\gamma'(t) = -K\gamma(t) + f(t), \ \ M_{ij} := \langle \varphi_i | \varphi_j \rangle, \ \ K_{ij} := -\langle \varphi_i | \mathcal{L} u_j \rangle, \ \ f_i(t) := \langle \varphi_i | f(t) \rangle, \tag{6.13}$$

which can be recognized as the standard discrete linear systems arising in finite element methods, with M and K the mass- and stiffness-matrix, respectively. Recall that since finite element techniques use non-overlapping elements to discretize the spatial domain and employ basis functions with localized support, the problem size scales as $m \sim p^d$ in d spatial dimensions, where p is the average number of elements per dimension. Thus, the curse-of-dimensionality arises from need to perform increasingly high-dimensional, albeit sparse, linear algebra in order to solve the linear system (6.13). The approach advocated in the following section, in contrast, overcomes the curse-of-dimensionality by representing the solution vector in terms of an autoregressive neural-network quantum state.

6.1.5 Neural-network quantum state implementation

In order to overcome the curse of dimensionality with respect to the spatial dimension d, which is inherent in the utilization of a d-dimensional mesh $\widehat{\Omega} \subset \Omega \subseteq \mathbb{R}^d$, we utilize stochastic estimation combined with autoregressive assumptions. In particular, we take inspiration from both the variational quantum algorithm introduced in [4] as well variational quantum Monte Carlo using autoregressive NQS [120, 60], by parametrizing the solution of the PDE as $u_\theta = \alpha\, \psi_\beta$ where ψ_β is a unit-normalized NQS with variational parameters β and $\alpha > 0$ is a scale factor, whose time-dependence must be determined from the evolution equations along with β. The variational parameters thus consist of the augmented parameter vector $\theta := (\log \alpha, \beta) \in \mathbb{R}^{p+1}$ where $\beta \in \mathbb{R}^p$ is an unconstrained vector representing the weights and biases of the neural network. Plugging the rescaled ansatz into the evolution

equations (6.4), one obtains an augmented system of first-order, non-linear ordinary differential equations determining the time-dependence of the augmented vector θ. The overlaps defining M and V are estimated using the VMC importance sampling technique, where the probability density is chosen to be the modulus-squared wavefunction $|\psi_\beta(x)|^2$.

6.2 Numerical Implementation

In this section, we provide detailed modeling assumptions required to implement the algorithm. After describing a mesh-based encoding of the state variable into a multi-qubit state, we then introduce the model architecture and computation mechanisms including the forward pass and the autoregressive sampling process. Pre-training is necessary in order to satisfy the initial condition of the PDE, and we adopt the standard approach [123] by updating the model iteratively on batches of randomly selected mesh points. Finally, we describe the stochastic estimation procedure used to evolve the variational trial state using McLachlan's variational principle.

6.2.1 Conversion to meshed form

Rather than working in the continuum, we assume spatial discretization of the function $u(t,\cdot)$ on a regular grid $\widehat{\Omega}$ embedded in the d-dimensional domain $\Omega = [a_1, b_1]$×...×$[a_d, b_d]$ with a total of 2^n grid points. Without loss of generality, we assume Ω is a regular hypercube satisfying $|b_1 - a_1| = ... = |b_d - a_d|$, and that the mesh size along each axis is $2^{n/d}$. In order to represent the state of the discretized field in terms of the state of an n-qubit system, we assign each computational basis state $|k_1, \ldots, k_n\rangle \in \mathbb{C}^{2^n}$ with $(k_1, \ldots, k_n) \in \{0,1\}^n$ to a linear index defined by $k = \sum_{i=1}^n k_i 2^i$ and then unravel the linear index to indices along each of d axes yielding a d-tuple $(\bar{k}_1, \ldots, \bar{k}_d) \in \{0, \ldots, n/d - 1\}^d$ defined by

$$\sum_{i=1}^{n} \bar{k}_i (2^{\frac{n}{d}})^{i-1} = k. \tag{6.14}$$

The mesh point $\boldsymbol{x}^k \in \widehat{\Omega}$ corresponding to the linear index $k \in \{0, 1, ..., 2^n - 1\}$ is thus

$$\boldsymbol{x}^k = (a_1 + \bar{k}_1 \Delta x, ..., a_d + \bar{k}_d \Delta x) \tag{6.15}$$

where $\Delta x = |b_1 - a_1|/2^{\frac{n}{d}}$. Henceforth, we do not distinguish the index k, the binary string $(k_1, \ldots, k_n)$ and the corresponding coordinate $\boldsymbol{x}^k \in \widehat{\Omega}$. The digitized representation of the

state of the PDE at time t is thus the following unnormalized n-qubit state

$$|u(t)\rangle = \sum_{k \in \widehat{\Omega}} u(\boldsymbol{x}^k, t)|k_1, \ldots, k_n\rangle \tag{6.16}$$

and likewise the parametrized approximation $|u_\theta\rangle$ of $|u(t)\rangle$ is given by,

$$|u_\theta\rangle = \sum_{k \in \widehat{\Omega}} u_\theta(\boldsymbol{x}^k)|k_1, \ldots, k_n\rangle \tag{6.17}$$

where $u_\theta(\cdot)$ is a function defined on $\widehat{\Omega} \subset \Omega$.

6.2.2 Autoregressive assumption and sampling

In order to obtain an expressive family of trial functions $u_\theta : \widehat{\Omega} \longrightarrow \mathbb{C}$ which furthermore admits an efficient stochastic estimation procedure, we express u_θ as multiple of a unit-normalized neural-network quantum state $\psi_\beta : \widehat{\Omega} \longrightarrow \mathbb{C}$ with variational parameters $\beta \in \mathbb{R}^p$,

$$u_\theta(\boldsymbol{x}^k) = \alpha\, \psi_\beta(k_1, \ldots, k_n) \ , \quad \|\psi_\beta\|_{\widehat{\Omega}} = 1 \ . \tag{6.18}$$

A simple method to ensure unit-normalization and efficient sampling is to follow the work of MADE [44] which exploited a masked version of a fully connected layer, where some connections in the computational path are removed in order to satisfy the autoregressive properties, In particular, if we assume a choice of variables such that the state variable is strictly positive[1] $u_\theta(x) > 0$, then a suitable choice of unit-normalized function is

$$\psi_\beta(k_1, \ldots, k_n) = \prod_{i=1}^{n} \sqrt{p_{\beta,i}(k_i|k_{i-1}, \ldots, k_1)} \ , \tag{6.19}$$

where $p_{\beta,i}(\cdot|k_{i-1}, \ldots, k_1)$ is the parametrized conditional probability distribution for the ith bit.

6.2.3 Pre-training

Before we perform the time evolution, initial variational parameters θ_0 must be selected in order to match the variational function $u_{\theta_0}(x)$ with the choice of initial condition $u_0(x)$ for $x \in \widehat{\Omega}$ in the spatial domain.

This can be achieved via optimization of the following objective function using stochastic

[1] This can be considered as a special case of the complex-valued case [120, 60].

gradient descent,

$$J(\alpha_0, \beta_0) = \left\| \alpha_0 |\psi_{\beta_0}\rangle - |u_0\rangle \right\|_{\widehat{\Omega}}^2 \ . \tag{6.20}$$

6.2.4 Evolution

In this section we discuss the details of the stochastic estimation of M and V necessary to evolve the augmented parameter vector $\theta = (\log \alpha, \beta) \in \mathbb{R}^{p+1}$. For simplicity we only consider the affine case

$$\mathcal{F}(t, x, u) = \mathcal{L}u(t, x) + f(t, x) \ . \tag{6.21}$$

Consider the decomposition,

$$M = \left[\begin{array}{c|c} M_{00} & M_{0,1:p} \\ \hline M_{1:p,0} & M_{1:p,1:p} \end{array}\right], \quad V = \left[\begin{array}{c} V_0 \\ \hline V_{1:p} \end{array}\right]. \tag{6.22}$$

It is convenient to introduce the following helper functions. In particular, define the Born probability distribution $\rho_\beta(x) \in [0, \infty)$, the wavefunction score $\sigma_\beta(x) \in \mathbb{C}^n$ and the local energy $l_\theta(t, x) \in \mathbb{C}$ as follows,

$$\rho_\beta(x) := |\psi_\beta(x)|^2 \ , \quad \sigma_\beta(x) := \frac{\nabla_\beta \psi_\beta(x)}{\psi_\beta(x)} \ , \qquad l_\theta(t, x) := \frac{(\mathcal{L}\psi_\beta)(x)}{\psi_\beta(x)} + \frac{f(t, x)}{\alpha} \ . \tag{6.23}$$

By straightforward calculus, we obtain

$$M_{00} = \alpha^2, M_{1:p,0} = M_{0,1:p} = \alpha^2 \operatorname{Re}\left[\mathop{\mathbb{E}}_{x\sim\rho_\beta} \sigma_\beta(x)\right], M_{1:p,1:p} = \alpha^2 \operatorname{Re}\left[\mathop{\mathbb{E}}_{x\sim\rho_\beta} \overline{\sigma_\beta(x)}\, \sigma_\beta(x)^T\right], \tag{6.24}$$

and

$$V_0 = \alpha^2 \operatorname{Re}\left[\mathop{\mathbb{E}}_{x\sim\rho_\beta} l_\theta(t, x)\right] \ , \qquad V_{1:p} = \alpha^2 \operatorname{Re}\left[\mathop{\mathbb{E}}_{x\sim\rho_\beta} \overline{\sigma_\beta(x)}\, l_\theta(t, x)\right]. \tag{6.25}$$

The expectation values over x are approximated using Monte Carlo sampling. In practice, the batch of randomly generated samples is represented in the form of a buffer $\mathcal{B} = \{x_i\}_{i=1}^B$ that stores the unique samples in the batch and a counter $\mathcal{C} = \{c_i\}_{i=1}^B$ that records the number of occurrences of each of the corresponding samples. Expectation values are then

approximated by sums of the following form,

$$\mathop{\mathbb{E}}_{x\sim\rho_\beta}[g(x)] \approx \frac{1}{\sum_{i=1}^{B} c_i} \sum_{i=1}^{B} c_i g(x_i). \tag{6.26}$$

Computation of per-sample gradient. The computation for both M and V requires direct access to the per-sample gradients $\{\nabla_\beta \psi_\beta(x_i)\}_{i=1}^{B}$, which is typically not directly accessible during a traditional backward pass by some deep learning software. In order to avoid inefficient forward and backward passes for each of the B samples, we utilize `BackPack` which collects the quantities necessary to compute the individual gradients and reuses them to compute the per-sample gradient without significant computational overhead.

Parallel extraction of matrix elements. Given a sample $x \in \widehat{\Omega}$, corresponding to a particular row-index of the operator $\widehat{\mathcal{L}} \in \mathbb{C}^{2^n \times 2^n}$, the calculation of the local energy $l_\theta(t, x)$ defined in (6.23) involves determining the nonzero entries of $\widehat{\mathcal{L}}$ in that row. Fortunately, the structure of the PDE problem ensures that the location and values of these entries can be determined in $O(\text{poly}(n))$ time.

We exploit CPU parallelization to determine the nonzero row entries for each sample in the batch. Note that the maximal number of nonzero entries per row is determined in advance; e.g. $2n^2 + 1$ for diffusion operator with Dirichlet boundary conditions.

Boundary Conditions. Boundary conditions are implemented on the grid using an appropriate choice of source function. In the case of Dirichlet conditions, for example, we choose the source function as follows

$$f(t, x) = u(t, x)\mathbb{1}_{\partial\Omega}(x) \tag{6.27}$$

where $\mathbb{1}_{\partial\Omega}$ denotes the indicator function for the set $\partial\Omega$. **Parameter update.** The network can be trained by incrementing the parameters using a Euler scheme in the direction $\delta\theta \in \mathbb{R}^{p+1}$ given by the solution of the following linear system

$$M\delta\theta = V\delta t, \tag{6.28}$$

where M, V are computed as discussed before. In practice, M is usually ill-conditioned due to the fact that it's essentially a sum of B rank one matrices. To stabilize the inverse operation, we consider the singular value decomposition $M = U\Sigma W^T$, and remove the diagonal values of Σ smaller than a small threshold $\epsilon = 10^{-12}$ to obtain Σ_r of size $r\times r$, as well as U_r, W_r of size $(p + 1)\times r$. The direction vector is therefore approximated by

$W_r \Sigma_r^{-1} U_r^T V \, \delta t$.

6.3 Diffusion with Gaussian Initializations

In this section, we show various numerical experiments demonstrating the convergence and run time analysis of our proposed approach. Here, we consider the d-dimensional heat equation for which $\mathcal{L} = D \nabla \cdot \nabla$ in (6.21) (diffusion constant set to $D = 0.1$) with either periodic or Dirichlet boundary conditions on $\partial\Omega$. For benchmarking purposes, we employ a finite difference method with the standard central difference scheme to discretize $\mathcal{L}$ (formulas given in Section 6.1.3) and forward Euler for time-stepping.

6.3.1 Experimental setup

For initialization we chose a discrete isotropic Gaussian, expressed in terms of the modified Bessel function $I_x(t)$ of integer order x,

$$u_0(x) = \prod_{i=1}^{d} e^{-t} I_{x_i}(t) \ , \tag{6.29}$$

where $t > 0$ is a parameter controlling the width of the Gaussian, and x ranges from 0 to 2^{n-1}. Pre-training was performed using Adam optimizer [68] for 50k iterations with batch size 128, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The learning rate was warm-started for the first $1/10$ total training iterations, then decayed by a factor of 10 at $3/7, 5/7$ of the total training iterations.

After completion of pre-training, the state was evolved for a total evolution time $T = 1$ using a step size of $\delta t = 5{\times}10^{-5}$ and a batch size of $B = 1024$ for Monte Carlo estimation on each iteration. The time development of the state was compared to the result of Euler time-stepping the initial condition (6.29) using the same step-size. Due to the exponential scaling of the matrix $\widehat{\mathcal{L}}$, we only establish this baseline for the number of qubits $n \leq 16$. All reported numerical results are averaged across exactly 5 seeds, each trained for 20k steps.

Table 6.1 reports the relative error between our method and forward Euler method over 20k iterations, for diffusion problem with Dirichlet boundary conditions. The relative error of the obtained solution is computed by comparing the norm of the space-time history of the variational state with the result of Euler development,

$$\text{error} := \frac{1}{T} \sum_{t \in \widehat{\mathcal{T}}} \frac{\big\| |u(t)\rangle - |u_{\gamma(t)}\rangle \big\|_{\widehat{\Omega}}}{\big\| |u(t)\rangle \big\|_{\widehat{\Omega}}} \ . \tag{6.30}$$

Table 6.1: Average relative error of our method in comparison with forward Euler method over 20k iterations. The proposed algorithm solution is compared with a Euler forward method for the diffusion equation over 20k iterations. For forward Euler method, the time step is 5×10^{-5} with a total time of 1. The relative error is computed as $\frac{1}{T}\sum_{t=1}^{T}\frac{\|u(t,x)-f(x;\theta_t)\|}{\|u(t,x)\|}$.

Operator	Boundary Condition	n/d	# of Dimensions d			
			1	2	3	4
Diffusion	Dirichlet	4	5.13×10^{-3}	7.92×10^{-3}	3.12×10^{-2}	1.47×10^{-1}
Diffusion	Dirichlet	5	2.91×10^{-3}	9.91×10^{-3}	7.24×10^{-2}	-

Table 6.2: Average running time over 2k iterations. The batch size used here is 500. Forward Euler method suffers from the exponential complexity, whereas our method, despite having an overhead running time, enjoys a polynomial scaling. Note that we cannot apply Euler for higher dimensions due to the memory constraint.

Operator	Method	n/d	# of Dimensions d								
			1	2	3	4	5	6	7	8	9
Diffusion	Euler	4	0.024	0.137	1.559	305.92	-	-	-	-	-
Diffusion	Euler	5	0.031	0.225	68.827	-	-	-	-	-	-
Diffusion	Ours	4	29.52	55.76	138.88	230.02	360.80	518.00	792.63	1214.09	1812.94
Diffusion	Ours	5	32.53	88.75	173.43	311.82	507.93	847.85	1355.16	2379.86	4123.61

In general, the overall solution obtained using our method matches well with that obtained using forward Euler method. Their discrepancy increases as the dimensionality of the problem increases; nonetheless, within a tolerable threshold (e.g., less than 10%). One remedy is to simply increase the batch size, for example, we show in Section 6.3.4 that the relative error improves from 15% to around 3% with a batch size that is ten times larger.

6.3.2 Running Time Analysis

Since we discretize the domain Ω using a grid of size $|\widehat{\Omega}| = 2^n$, the time complexity of forward Euler method, expressed in terms of n is $O(T\times2^{2n})$, where T is the number of iterations. The proposed VMC algorithm, in contrast, scales as $O(TB\,\mathrm{poly}(n))$, where B is the batch size. In more detail, the forward pass scales as $O(TBn^2)$, and the sampling is $O(TBn^3)$, due to the sequential nature of the auto-regressive sampling process. This polynomial scaling comes at a price of the approximate nature of the time evolution step, and the implicit access to entries of the state vector compared to the forward Euler method which offers $O(1)$ lookup to entries of the state.

In Table 6.2, we report the average running time of both forward Euler method and our method for 2k iterations (one-tenth of the total running time). The batch size B of our method is fixed to be 500. Note that we cannot apply Euler for higher dimensions

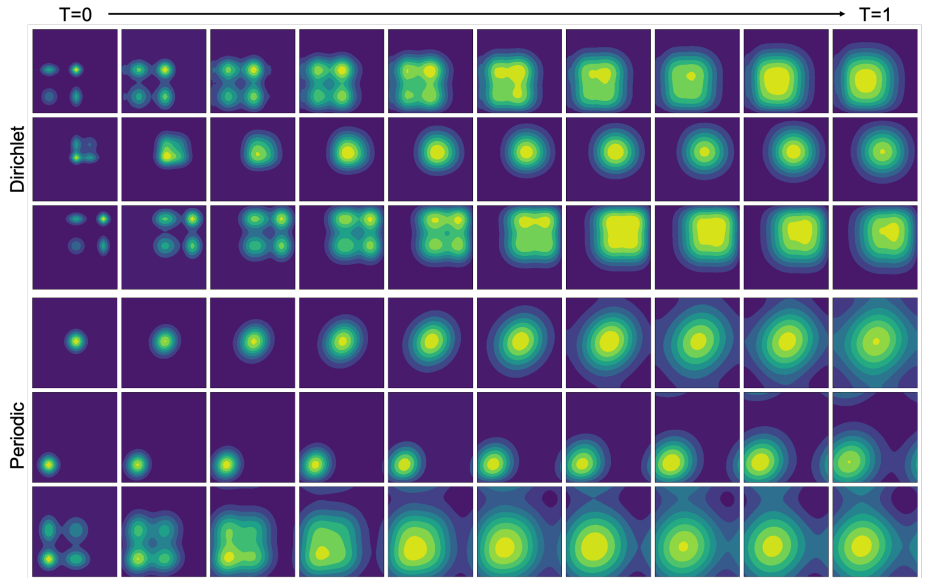


Figure 6.1: Snapshots of the evolution obtained using our algorithm for diffusion equation with Dirichlet and periodic boundary conditions and different choices of initialization.

due to the memory constraint. Although forward Euler method is effective for small-scale problems, its complexity suffers from the exponential growth with respect to dimension d. The computational cost of our method originates from four sources: sampling, forward pass, per-sample gradient computation (backward pass), and extraction of matrix element information. All these sources contribute to the overhead time that causes our method to run slower in comparison with respect to forward Euler method for problems in lower dimensions. However, as the dimensionality increases, the run time of our method grows only at a polynomial rate. It can be seen from Table 6.2 that our method is already faster than forward Euler for problem sizes characterized by $n = 16$ qubits.

6.3.3 Convergence Visualization

We provide snapshots of our method for 2D diffusion problems with periodic boundary conditions over 20k iterations. We run our algorithm with five distinct initializations and record the snapshot every 2k iterations. In particular we do not employ any regularization techniques to enforce that the solution satisfies the boundary condition. It can be observed in Figure 6.1 that our method successfully obeys the periodic boundary conditions.

6.3.4 Ablation Study on Batch Size

Recall that M and V are approximated with Monte Carlo sampling using batches of unique samples. Intuitively, a larger batch size yields a better approximation to the exact expectation

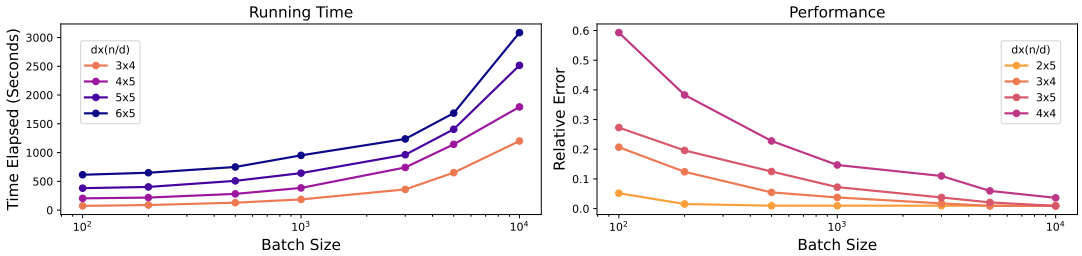


Figure 6.2: We report the running time and the average relative error with the forward Euler method over various batch sizes, for qubit sizes $n = d \times \frac{n}{d}$, where d is the dimensionality of the problem. The running time grows with respect to the batch size and the problem size. On the other hand, the performance is greatly improved when training with larger batch sizes.

value, thereby providing more accurate model updates. In this section, we study the effect of batch size on the performance of our method. In the LHS of Figure 6.2, the running time of our algorithm increases for both larger problem sizes and batch sizes. Note that the actual running time does not grow linearly with respect to the batch size in the plot due to the cache and parallelization. In the RHS of Figure 6.2, we report the average relative error between the forward Euler method and the VMC method with various batch sizes. Given a fixed problem size (e.g., 4 dimensions with 4 qubits per dimension), we observe a performance improvement by increasing the batch size, which verifies our hypothesis that increasing the batch size does effectively improve the performance. Given a fixed batch size, our method performs worse as the dimensionality of the problem increases. This result implies that we need a larger batch size to guarantee good performance for problems in higher dimensions.

6.4 Option Pricing

In this section, we explain how our results can be used to price options. We start with the well-known Feynman-Kac representation theorem and then describe how the numerical results obtained in Section 6.3 can be applied to the multi-dimensional Black-Scholes model.

6.4.1 The Feynmam-Kac representation theorem and option pricing

The Feynman-Kac stochastic representation formula links between a parabolic PDE and stochastic differential equations. Namely, consider a PDE of the form:

$$\begin{cases} \frac{\partial u}{\partial t}(t,x) + \mu(t,x) \cdot \nabla_x u(t,x) + \frac{1}{2}\text{trace}[\sigma^T H \sigma](t,x) - r(t,x)u(t,x) + f(t,x) = 0, \\ u(T,x) = \Phi(x), \qquad\qquad (t,x) \in [0,T] \times \mathbb{R}^d, \end{cases}$$

where $\mu : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$, $\sigma : [0,T] \times \mathbb{R}^d \to \mathbb{R}^{d\times d}$, $r, f : [0,T] \times \mathbb{R}^d \to \mathbb{R}$, and $\Phi : \mathbb{R}^d \to \mathbb{R}$ are given measurable functions and H stands for the Hessian matrix of u:

$$H_{ij} = \frac{\partial^2 u}{\partial x_i \partial x_j}.$$

Then, under a second order integrability condition, the solution to this equation takes the form:

$$u(t,x) = \mathbb{E}\Big[\int_t^T e^{-\int_t^\tau r(s,X_s)ds} f(\tau, X_\tau)d\tau + e^{-\int_t^T r(s,X_s)ds}\Phi(X_T)\Big| X_t = x\Big], \qquad (6.31)$$

where $(X_t)_{t\in[0,T]}$ is the solution to the following stochastic differential equation with $(W_t)_{t\in[0,T]}$ being a d-dimensional standard Wiener process:

$$\begin{cases} dX_t = \mu(t, X_t)dt + \sigma(t, X_t)dW_t, \\ X_t = x. \end{cases}$$

One of the many applications of this representation is in derivative pricing. Consider multi-asset with price dynamics $(S_t)_{t\in[0,T]}$ given by the multi-dimension geometric form:

$$\begin{cases} dS_t = \text{Diag}[S_t]\mu(t, S_t)dt + \text{Diag}[S_t]\sigma(t, S_t)dW_t, \\ S_0 = s_0 > 0, \end{cases}$$

where Diag$[x]$ is a diagonal matrix with the vector x on the diagonal, W is d-dimensional standard Wiener process, $\mu : [0,T] \times [0,\infty) \to \mathbb{R}^d$ and $\sigma : [0,T] \times [0,\infty) \to \mathbb{R}^{d\times d}$ are measurable and satisfy basic conditions that lead to a unique strong solution to the stochastic differential equation above. Consider also the interest rate $r : [0,T] \times [0,\infty) \to \mathbb{R}$. A *risk-neutral measure* is a probability measure $\mathbb{Q}$, such that the discounted price of the asset $(e^{-\int_0^t r(s,X_s)ds}X_t)_{t\in[0,T]}$ is a martingale under $\mathbb{Q}$. The dynamics of the multi-asset under $\mathbb{Q}$ are given by:

$$\begin{cases} dS_t = r(t, S_t)S_t dt + \text{Diag}[S_t]\sigma(t, S_t)dW_t^{\mathbb{Q}}, \\ S_0 = s_0 > 0, \end{cases}$$

where $W_\cdot^{\mathbb{Q}} := W_\cdot - \int_0^\cdot (\mu(t, S_t) - r(t, S_t))dt$ is a standard Wiener process under $\mathbb{Q}$. Then, the price of a simple contingent claim with terminal payment $\Phi(S_T)$ at any given time $t \in [0,T]$, when the multi-asset prices are $S_t = s$ is given by $u(t,s)$, which satisfies the

PDE:

$$\begin{cases} \frac{\partial u}{\partial t}(t,s) + r(t,s)s \cdot \nabla_x u(t,s) + \frac{1}{2}\text{trace}[\sigma^T H \sigma](t,s) - r(t,s)u(t,s) = 0, \\ u(T,s) = \Phi(s), \qquad\qquad (t,s) \in [0,T] \times (0,\infty)^d, \end{cases}$$

and can be expressed as the conditional $\mathbb{Q}$-expectation as follows:

$$u(t,s) = \mathbb{E}^{\mathbb{Q}}\Big[e^{-\int_t^T r(u,S_u)du}\Phi(S_T)\Big|S_t = s\Big].$$

The most celebrated example is the Black-Scholes model, where the dynamics of the underlying asset follows a geometric Brownian motion and the interest rate is fixed. In dimension 1, the associated PDE admits an explicit solution, which is known as the *Black-Scholes formula*. Aside from the fact that the solution of the PDE is the contingent claim price, its partial derivative $\partial u/\partial x$ is used in order to construct a *delta-hedging* portfolio. Furthermore, its partial derivatives, known as the *Greeks* are used to construct a robust portfolio against small changes, when moving from the continuous- to the discrete-time world.

6.4.2 Option pricing in Black-Scholes model

We now demonstrate how our algorithm applied to the multidimensional heat equation can be used to price options. For this we consider the Black-Scholes model that consists of a risk-free asset with a constant risk-free return $r > 0$ and d risky assets whose dynamics are given by

$$dS_t^i = \mu_i S_t^i dt + \sigma_i S_t^i dW_t^i, \qquad i = 1, \dots, d.$$

The parameters μ_i and σ_i, $i = 1, \dots, d$ are constants, and $\{W^i\}_{i=1}^d$ are Wiener processes with quadratic covariation $[W_t^i, W_t^j] = \rho_{ij}t$.

Consider further a European option, whose payment at (the predetermined) expiry time $T > 0$ is $\Psi(S_T^1, \dots, S_T^d)$, for some measurable function Ψ. Let V be the conditional price of this option, i.e., $V(t, x_1, \dots, x_d)$ is the price for the option at time t, given that $S_t^i = x_i$ for $i = 1, \dots, d$. It is well-known that V satisfies the following Black-Scholes PDE:

$$\begin{cases} \frac{\partial V}{\partial t} + \sum_{i=1}^d r x_i \frac{\partial V}{\partial x_i} + \frac{1}{2}\sum_{i=1}^d \sigma_i^2 x_i^2 \frac{\partial^2 V}{\partial x_i^2} + \sum_{i \neq j} \frac{1}{2}\rho_{ij}\sigma_i\sigma_j x_i x_j \frac{\partial^2 V}{\partial x_i \partial x_j} - rV = 0, \\ V(T, x_1, \dots, x_d) = \Psi(x_1, \dots, x_d). \qquad\qquad (t,x) \in [0,T] \times (0,\infty)^d \end{cases}$$

Following Guillaume [50], we may reduce this n-dimensional equation to the n-dimensional

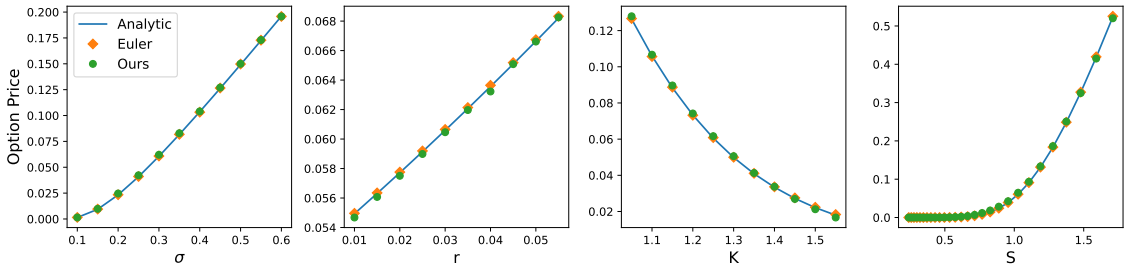


Figure 6.3: Ablation study on volatility σ, interest rate r, strike price K and initial price. We fix a base setting with hyper-parameters $\sigma = 0.3, r = 0.03, K = 1.25$, and run our algorithm on each setting with only one hyper-parameter deviated from the base setting. In addition, we plot the wave function under the base setting. We compare our solution at the execution time T versus forward Euler method and the corresponding analytic ground truth. Our method is robust under all settings and achieves satisfactory performance.

standard heat equation. To this end, set

$$u(t, y_1, \ldots, y_n) = V(T - t, e^{\sigma_1 y_1}, \ldots, e^{\sigma_n y_n}) e^{\sum_{i=1}^{n} -a_i \sigma_i y_i - bt},$$

where a_i and b satisfy the following system of equations:

$$\sum_{i=1}^{d} a_i \left(r - \frac{\sigma_i^2}{2} \right) + \frac{1}{2} \sum_{i=1}^{d} a_i^2 \sigma_i^2 + \sum_{i \neq j} \frac{1}{2} \rho_{ij} \sigma_i \sigma_j a_i a_j - r - b = 0,$$

$$r - \frac{\sigma_i^2}{2} + a_i \sigma_i^2 + \sum_{j \neq i} \rho_{ij} \sigma_i \sigma_j a_j = 0, \qquad i = 1, \ldots, d.$$

Then, u satisfies the following heat equation

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{1}{2} \sum_{i=1}^{d} \frac{\partial^2 u}{\partial y_i^2} + \sum_{i \neq j} \frac{1}{2} \rho_{ij} \frac{\partial^2 u}{\partial y_i \partial y_j}, & (t, y) \in [0, T] \times \mathbb{R}^d, \\ u(0) = \Psi(e^{\sigma_1 y_1}, \ldots, e^{\sigma_d y_d}) e^{-\sum a_i \sigma_i y_i}. \end{cases}$$

with $u(0) = \Psi(e^{\sigma_1 y_1}, \ldots, e^{\sigma_d y_d}) e^{-\sum a_i \sigma_i y_i}$.

6.4.3 Numerical Experiments

In this section, we apply our algorithm to option pricing in Black-Scholes model across different settings using the numerical solution to the heat equation and the translation from the heat equation to the Black-Scholes equation from the previous subsection. We test the performance of our algorithm as well as show a calculation of the option price for higher dimensions.

In Figure 6.3 and Table 6.4 we provide examples to test the performance of our algorithm.

Option Type	Payoff Function at expiry $\Psi(s)$
1D Call	$\max(s-K,0)$
Basket Call	$\max(\sum w_i s_i - K, 0)$
Basket Put	$\max(K - \sum w_i s_i, 0)$
Rainbow Max Call	$\max(\max s_i - K, 0)$
2D Spread Put	$\max(K-(s_1-s_2),0)$

Table 6.3: Payoff functions for our experiments. We consider basket call and put, Rainbow max call, and spread put options.

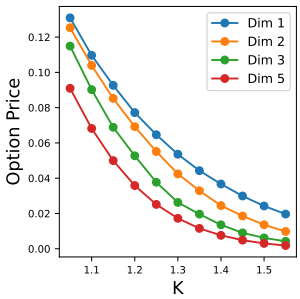


Figure 6.4: Ablation study on dimensionality, following the base settings in Figure 6.3, $\sigma = 0.3, r = 0.03, K = 1.25$.

In both we include 1D examples for the price of a European call option, whose value at the expiry time T is $V(T,s) = \Psi(s) = \max\{s-K,0\}$, where K is a predetermined constant, called the strike price. We vary the volatility σ, strike price K, interest rate r, expiry time T, and initial price of the stock S. In Figure 6.3, we compare against the forward Euler method and the ground truth *Black-Scholes formula*, which admits an analytical solution in 1D. Specifically, it is given by

$$V(t,s) = N(d_+)s - N(d_-)Ke^{-r(T-t)}, \tag{6.32}$$

where N is the cumulative distribution function of the standard normal distribution and

$$d_\pm = \frac{\ln \frac{s}{K} + (r \pm \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}.$$

Our method is robust under all settings and achieves satisfactory performance. In Table 6.4 we compare our accuracy against the forward Euler method, where we use the same analytical solution. Although the performance of our algorithm is inferior to that of Euler, it still achieves good accuracy and the margin can be treated as a price to pay for reducing the exponential scaling down to a polynomial one. Our approach is valuable for high dimensions, where other methods, such as the forward Euler method, suffer from the curse of dimensionality. Table 6.4 also includes 2D examples with four different options: *Basket European call* and *put*, *rainbow max European call*, and *spread European put*, whose payoffs are listed in Table 6.3. Note that as we don't have an analytical solution for this case, the relative errors with respect to Euler solutions are reported instead. In Figure 6.4, we provide a graph for the price of a basket European call option with up to five underlying stocks as a function of the strike price K. As expected the prices are convex with K.

Note that the Black-Scholes PDE lives on the positive orthant of $\mathbb{R}^d$ while its translation

Table 6.4: List of experiments for the application of our algorithm to Black–Scholes equation. The hyper-parameters for the experiments are listed. We compute the relative error of our method (**Ours**) at expiration time T with analytical ground truth in the 1D case and Euler solution in the 2D case, respectively. For 1D, we also report the relative error of the forward Euler method with respect to the analytical ground truth (**Euler**) for comparison. Our algorithm achieves robust performance across various settings.

Problem	Initial & Boundary Cond.	d	D	T	r	K	σ	Ours	Euler
Black–Scholes 1D	1D CALL	1	-	1	0.03	1.25	0.3	0.011781	0.002494
Black–Scholes 1D	1D CALL	1	-	1	0.03	1.25	0.1	0.032792	0.000930
Black–Scholes 1D	1D CALL	1	-	1	0.03	1.25	0.2	0.017272	0.002389
Black–Scholes 1D	1D CALL	1	-	1	0.03	1.25	0.4	0.011560	0.001392
Black–Scholes 1D	1D CALL	1	-	1	0.03	1.05	0.3	0.086764	0.001750
Black–Scholes 1D	1D CALL	1	-	1	0.03	1.15	0.3	0.013365	0.002521
Black–Scholes 1D	1D CALL	1	-	1	0.03	1.35	0.3	0.012260	0.000328
Black–Scholes 1D	1D CALL	1	-	1	0.03	1.45	0.3	0.012626	0.000845
Black–Scholes 1D	1D CALL	1	-	1	0.01	1.25	0.3	0.010436	0.002538
Black–Scholes 1D	1D CALL	1	-	1	0.02	1.25	0.3	0.010599	0.002515
Black–Scholes 1D	1D CALL	1	-	1	0.04	1.25	0.3	0.014192	0.002477
Black–Scholes 1D	1D CALL	1	-	1	0.05	1.25	0.3	0.017423	0.002463
Black–Scholes 1D	1D CALL	1	-	0.5	0.03	1.25	0.3	0.022481	0.018385
Black–Scholes 1D	1D CALL	1	-	1.5	0.03	1.25	0.3	0.012049	0.014881
Black–Scholes 2D	2D BASKET CALL	2	0.1	1	0.03	1.25	0.3	0.053477	-
Black–Scholes 2D	2D BASKET PUT	2	0.1	1	0.03	1.25	0.3	0.043926	-
Black–Scholes 2D	2D RAINBOW MAX CALL	2	0.1	1	0.03	1.25	0.3	0.057949	-
Black–Scholes 2D	2D SPREAD PUT	2	0.1	1	0.03	1.25	0.3	0.031574	-

to the heat equation lives on $\mathbb{R}^d$. Thereby, all numerical algorithms, including ours, require artificial truncation of the domain. We choose the hypercube domain to be $[s_l, s_u]^d = [Ke^{-3\sigma_i\sqrt{T}}, Ke^{3\sigma_i\sqrt{T}}]^d$. This choice implies that s_l is small (close to $0+$) and s_u is large (close $+\infty$). On the faces of the hypercube, we use the time-discounted payoff functions, as they are reasonably accurate approximations of boundary values of the options considered. Given the number of qubits n and the hypercube input domain for the heat equation $[L_l, L_u]^d$, which is approximately $[-5, 5]^d$, the mesh size of each axis is $(L_u - L_l)/(2^{n/d} + 1)$.

6.5 Conclusions

In summary, we introduced a generalization of McLachlan's variational principle applicable to generic time-dependent PDEs as well as a quantum-inspired training algorithm based on neural-network quantum states which can be used to perform approximate time evolution in high dimensions, overcoming the curse-of-dimensionality. Although we focused on a mesh-based formulation in which the quantum state vector is represented by n qubits, it

is clear that the mesh is not mandated by the formulation and it would be very interesting to pursue meshless variants based on continuous-variable neural-network quantum states including normalizing flows [129] and to address non-trivial boundary conditions. There exist a number of directions in which the results in this section can be potentially improved. Since we only considered a first-order Euler approximation of the ODE (6.4) it would be natural to incorporate high-order time stepping schemes (e.g., Runge-Kutte methods). As an alternative, it would be interesting to pursue a direct solution of the discrete-time dynamical system (6.3) which has proven successful in both the VMC [51] and VQA [7] literature.

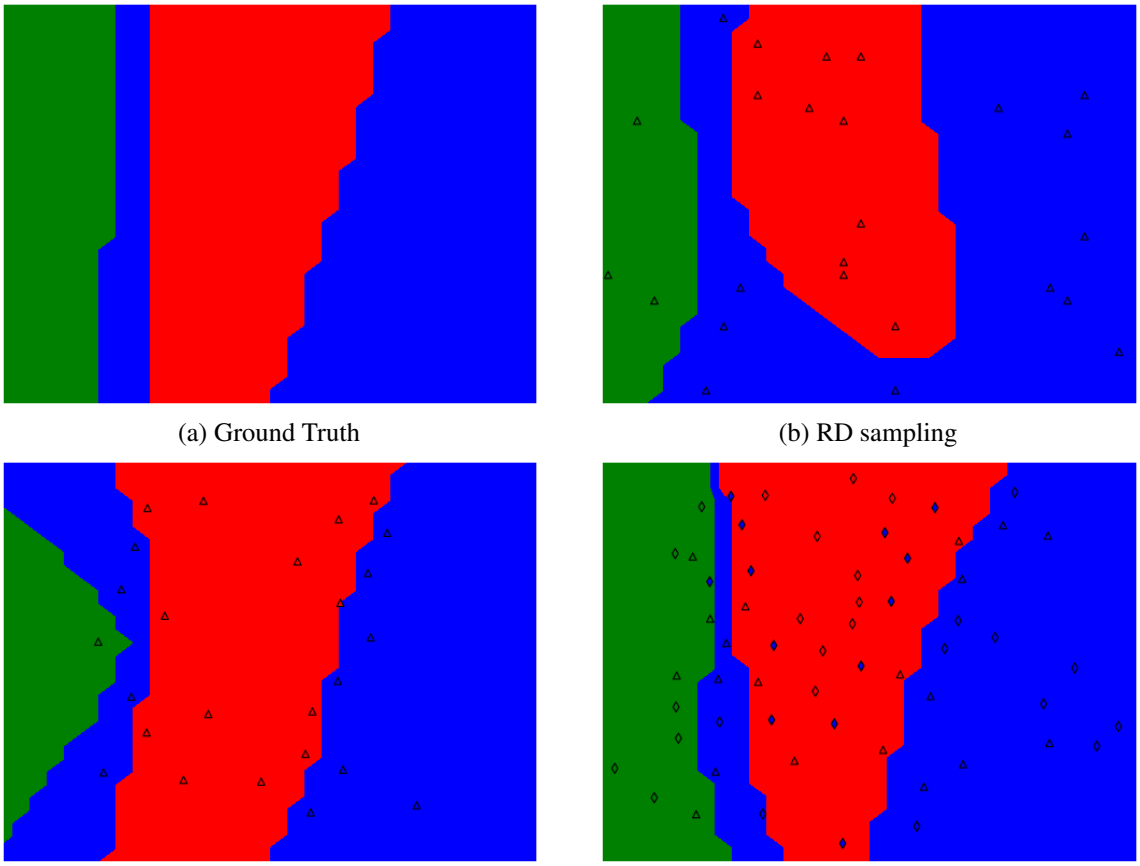
Chapter 7

Multi-Fidelity Active Learning in High Dimensional Space

Many tasks in scientific and engineering applications require a large number of labeled instances, which are very difficult, time-consuming, or expensive to obtain. The labeling source can be complex black-box simulation [21, 89] or sophisticated human expert [153]. To make the problem tractable, practitioners often replace the source with a surrogate model that generates low-cost approximations. In this chapter, we investigate the approach that constructs the surrogate through data-fitting [103]. Our work is closely related to Surrogate-Based Optimization (SBO) [39], where the objective is to locate the global maximum of an underlying expensive-to-evaluate function. We are interested in estimating the decision boundary of an unknown classifier, which has applications arising in several domains, such as Reliability Analysis [10, 14], Aeroelastic modeling [32, 84], Classification-Based Black-Box Optimization [143, 146], Computerized electrocardiogram (ECG) interpretation [58], skin cancer classification [34], drug classification [110], *etc*.

A large dataset is required to train high-performance surrogates, however, generating even a small training set from the source might be extremely expensive [82]. Active learning [116] attempts to overcome the bottleneck by only selecting and labeling the most informative instances through some information measure, *i.e*., the acquisition function. The algorithm aims to achieve high accuracy using as few labeled instances as possible, thereby minimizing the total data generation cost.

In many applications, besides the expensive-to-evaluate high-fidelity (HF) source, we also have access to cheaper approximations evaluated by different means. For example, medical image classification data is expensive because it requires expert knowledge from physicians; however, cheaper but occasionally incorrect labels can also be obtained from medical residents. In dynamical-system simulation, examples of low-fidelity (LF) sources include inexact solutions arising from the early termination of an iterative method, numerical simulations characterized by simplifying physical assumptions, or a coarse discretization.



(a) Ground Truth

(b) RD sampling

(c) AL sampling

(d) MFAL sampling

Figure 7.1: Visualizations of surrogates trained with **(b)** Latin Hypercube random (RD) sampling, **(c)** Active Learning (AL) sampling, and **(d)** our Multi-Fidelity Active Learning (MFAL) sampling. The ground truth is given in **(a)**, which is a 32×32 contour plot of three classes, indicated by red, blue, and green, where each point in the plot is a classification result from 2D Cahn-Hilliard simulation parametrized by a temperature coefficient (horizontal axis) and a diffusion coefficient (vertical axis). The experiments are done in the low-data regime: RD and AL make 25 HF queries, whereas MFAL makes 20 HF queries and 40 LF queries, represented by triangles and diamonds respectively. Although LF queries sometimes provide inaccurate information, our algorithm can leverage them effectively and improve the surrogate's performance significantly with fewer HF queries. See Section 7.3.2 for more details.

We refer to these as different information sources (ISs), where each IS may have its advantage in estimating the true value of a quantity of interest, at a given cost. Multiple ISs provide extra flexibility to establish the optimal cost-efficient strategy for data sampling and labeling.

One branch of the multi-fidelity modeling literature focuses on predicting statistics of the information sources using sampling approaches, where the expensive multi-fidelity Monte Carlo methods are proposed to reduce the variance of the classical Monte Carlo estimator [45, 54, 48]. Another branch considers the scenario where the multi-fidelity models admit a multi-level hierarchy structure, *e.g.*, the hierarchies of fine- and coarse-grid discretizations,

shifting most of the sampling budget onto the cheap lower-fidelity models but correcting or co-training with a few from the expensive high-fidelity model [132, 53, 67]. On the other hand, for closely related topics such as multi-fidelity optimization, majority of the work uses Gaussian process (GP) surrogates for regression tasks [40, 131, 72, 73, 101]. However, the Gaussian likelihood assumption used in the formulation of GP is not appropriate for modeling discrete class labels, and no analytic inference scheme can be derived for training or inferencing [105]. Besides, GPs are known not to be robust in providing uncertainty estimates for high-dimensional inputs with large training sets, especially in classification tasks [134, 63].

In this chapter, we propose a Multi-Fidelity Active Learning (MFAL) algorithm to maximize the cost-efficiency in data collection for surrogate training. More precisely, given the ability to interactively make label queries to HF source at a higher cost or several LF sources at lower costs, our goal is to optimally select and label the instances in the input domain such that the classifier with the highest performance can be trained under a fixed labeling budget. Our work follows a direct data-driven approach that learns the correlations across the instance and fidelity domains, which, together with a generalized multi-fidelity acquisition function, provides global uncertainty estimates for instance sampling and querying.

Our contributions lie as follows. First, we propose a simple learning framework with neural networks that is capable of handling large-scale high-dimensional data generated by an arbitrary number of ISs, and is significantly more efficient than the GP approach [109]. Second, we propose MFAL, an augmented active learning algorithm that selects both the instance and the source to query at every iteration. MFAL applies to any choice of acquisition function in the active learning literature for probabilistic models and does not make any assumptions on the LFs other than there may exist some unknown correlations among the HF and LFs. Our experiments show that MFAL out-performs Latin Hypercube random sampling and Active Learning sampling by a noticeable margin.

7.1 Background

Multi-Fidelity Bayesian Optimization with Gaussian Processes has been studied extensively in the literature. [73, 131, 83] propose different designs of cost-sensitive acquisition functions to utilize cheap auxiliary task in the minimization of an expensive primary task. [101, 72, 109] use Co-Kriging methods to fuse the information from multi-fidelity sources.

For the application of GP models on physics-based simulation and engineering calibration, Parish and Carlberg [97] proposes a Time-Series Machine-Learning Error Modeling (T-MLEM) for modeling the error incurred by approximate solutions to parameterized dy-

namical systems. Perdikaris et al. [99] develops a framework for multi-fidelity information fusion through stochastic autoregressive schemes and frequency-domain machine learning algorithms, on benchmark problems up to 10^5 input dimensions and 10^5 training points. Perdikaris et al. [98] and Sarkar et al. [114] apply their multi-fidelity Gaussian process model on applications such as stochastic burgers equation, stochastic incompressible flow, design optimization of compressor rotor, *etc*.

For classification tasks, Zhang and Chaudhuri [149] considers an active learning algorithm from a weak and a strong labeler with probabilistic models and analyzes its statistical consistency. Dribusch et al. [32] introduces a multi-fidelity approach for the construction of explicit boundaries with SVM; the methodology defines an envelope encompassing a lower fidelity boundary in order to limit the number of high fidelity calls. Donmez and Carbonell [31] builds a decision-theoretic framework with logistic regression that casts the multi-oracle optimization problem as a utility optimization problem subject to a budget constraint.

Existing methods in the literature either assume the hierarchical relationships among the sources or rely on the heuristic design of the cost-weighted acquisition function that is either problem-dependent or practically intractable. We propose a flexible Multi-Fidelity Active Learning algorithm that is applicable to any designs of acquisition function, without any restrictive assumptions on the LFs.

7.2 Algorithm

7.2.1 Problem Statement and Notations

Given a hypothesis class $\mathscr{H}$, an instance domain $\mathcal{X}$, and the ability to interactively make label queries to a high-fidelity (HF) source at a higher cost or several low-fidelity (LF) sources at lower costs, the objective is to train a high-performance classifier in $\mathscr{H}$, by selecting an optimal dataset under a fixed budget. In this work, the instance domain $\mathcal{X}$ is assumed to be a hypercube $[0,1]^n$, for some integer $n > 1$. The set of ISs is denoted by $\mathcal{G} = \{g_\lambda\}_{\lambda \in \Lambda}$, where the fidelity space Λ is finite, specified by the index set $\{0, 1, ..., M\}$. Each IS is a function that maps from $\mathcal{X}$ to the discrete label set $\mathcal{Y} = \{1, 2, ..., C\}$. The surrogate model is a neural network denoted by f. To fix the idea, we assume there is one HF source g_0 that we would like to approximate, and $\{g_\lambda\}_{\lambda=1}^{M}$ are the LF sources to help with the approximation. However, this assumption is not required for our proposed algorithm.

Algorithm 4 Active Learning

Input: Query budget N, source g, instance domain $\mathcal{X}$, acquisition function $\mathcal{A}$
Initialize: Train surrogate with initial dataset $\mathcal{D}$.
while Query budget is not exhausted **do**
  Sample m query candidates $\{\boldsymbol{x}^{(j)}\}_{j=1}^{m}$ from $\mathcal{X}$.
  Compute the acquisition scores $\{\mathcal{A}(\boldsymbol{x}^{(j)})\}_{j=1}^{m}$.
  Find the query instance $\boldsymbol{x}^*$ with the maximum score.
  Query g for $\boldsymbol{x}^*$ to obtain label y^*.
  Update dataset $\mathcal{D}$ with $(\boldsymbol{x}^*, y^*)$.
  Re-Train surrogate with $\mathcal{D}$.
end while

7.2.2 Active Learning

We consider the scenario of pool-based sampling [76], where a small set of labeled data is available, sampled from a large static domain $\mathcal{X}$ of unlabeled instances. Instances are greedily drawn from $\mathcal{X}$, according to an acquisition function which evaluates the informativeness of unlabelled instances. There have been many approaches on the formulation of acquisition function, the most favorite two are Uncertainty sampling [76] and Query-By-Committee (QBC) algorithm [117]. A comprehensive review and benchmarks on active learning can be found in [116, 144].

Entropy [118] is an information-theoretic measure that represents the amount of information needed to "encode" a distribution, with an interpretation as a measure of uncertainty or impurity in machine learning. We adopt the general uncertainty sampling strategy using entropy as an acquisition function $\mathcal{A}$ for uncertainty measure:

$$\mathcal{A}(\boldsymbol{x}) = \mathrm{H}(Y|\boldsymbol{x}) = -\sum_{y\in\mathcal{Y}} p(y|\boldsymbol{x}) \log p(y|\boldsymbol{x}),\ \boldsymbol{x} \in \mathcal{X}. \tag{7.1}$$

In Bayesian's framework, the predicted probability $p(y|\boldsymbol{x})$ is the posterior distribution on the class label y with a training dataset $\mathcal{D}$:

$$p(y|\boldsymbol{x}, \mathcal{D}) = \int_{\theta} p(y|\boldsymbol{x}, \theta) p(\theta|\mathcal{D}) d\theta. \tag{7.2}$$

However, this integral cannot be evaluated in a closed form for large scale neural networks, *i.e.*, very high-dimensional parameter variable θ. One practical approach [122] is to approximate the parameter posterior $p(\theta|\mathcal{D})$ by some tractable distribution, and approximate the integral through sampling. We use Dropout [125], a technique that induces stochasticity to the parameters, by randomly dropping units from the neural network. More precisely,

consider a random binary mask variable M of the model parameter θ, with each entry following Bernoulli(p) for some probability $p \in (0, 1)$. We approximate the integral in Eq. (7.2) by the average over T samplings:

$$p(y|\boldsymbol{x}, \mathcal{D}) = \frac{1}{T} \sum_{i=1}^{T} p(y|\boldsymbol{x}, \theta^i), \tag{7.3}$$

where $\theta^i = \theta \odot \boldsymbol{m}^i$, $\{\boldsymbol{m}^i\}_{i=1}^T$ are realizations of M with entries drawn from Bernoulli(p). The probabilities $p(y|\boldsymbol{x}, \theta^i)$ are computed from the logits vector $\boldsymbol{\eta} = f(\boldsymbol{x}|\theta^i)$ and the softmax function $p(y = j|\boldsymbol{x}; \theta^i) = e^{\boldsymbol{\eta}_j} / \sum_{c=1}^{C} e^{\boldsymbol{\eta}_c}$. It's been shown that neural networks with Dropout applied after every weight layer is mathematically equivalent to GP [41].

At every iteration, the maximizer $\boldsymbol{x}^* = \text{argmax}_{\boldsymbol{x} \in \mathcal{X}} \mathcal{A}(\boldsymbol{x})$ is selected as the query instance. However, for continuous instance domain $\mathcal{X}$, searching the maximizer is itself an optimization problem. In practice, one samples m query candidates from $\mathcal{X}$ randomly and select the one with a maximum score by $\mathcal{A}$. The algorithm is summarized in Algorithm 4, where the query budget is the total number of queries allowed.

A naive execution of Algorithm 4 sometimes yields sampled instances that are geometrically too close to each other. Besides, the surrogate f might be over-confidence in its predictions over some regions of the instance domain $\mathcal{X}$ which are in fact incorrect; these regions will never get the attention from the entropy-based method. To address this problem, we occasionally explore the unexplored region by solving

$$\mathcal{A}(\boldsymbol{x}) = \min_{\boldsymbol{x}' \in \mathcal{D}_{\boldsymbol{x}} \bigcup \partial\mathcal{X}} d(\boldsymbol{x} - \boldsymbol{x}'), \tag{7.4}$$

where $\mathcal{D}_{\boldsymbol{x}}$ is the collection of the sampled instances and $\partial\mathcal{X}$ is the boundary of $\mathcal{X}$. The distance metric d is usually taken to be the Euclidean norm $\|\cdot\|_2$.

Finally, we remark that since our instance domain $\mathcal{X}$ is continuous, we do not explicitly remove the sampled instance from the domain like what classical active learning algorithm does, as the probability that the same instance is sampled more than once is zero. Instead, Eq. (7.4) can be considered as a regularizer to Eq. (7.1), with an interpretation of discouraging the clustering of sampled data. We adopt this trick for some of our experiments.

7.2.3 Multi-Fidelity Active Learning

The sampling efficiency can be further improved if additional less accurate but cheap ISs are available. For example, suppose we would like to know the label $g_{\lambda_1}(\boldsymbol{x})$ of the instance $\boldsymbol{x}$ at

Algorithm 5 Multi-Fidelity Active Learning

Input: Query budgets $\{N_i\}_{i=0}^{M}$, sources $\{g_i\}_{i=0}^{M}$, instance domain $\mathcal{X}$, acquisition function $\mathcal{A}$
Initialize: Train surrogate with initial dataset $\mathcal{D}$.
while Query budgets are not exhausted **do**
  Sample m instances $\{\boldsymbol{x}^{(j)}\}_{j=1}^{m}$ from $\mathcal{X}$ randomly.
  Sample m fidelities $\{\lambda^{(j)})\}_{j=1}^{m}$ from Λ with scaled probabilities.
  Compute the acquisition scores $\{\mathcal{A}(\boldsymbol{x}^{(j)}, \lambda^{(j)})\}_{j=1}^{m}$.
  Find $\boldsymbol{x}^*, \lambda^*$ with the maximum score.
  Query the source g_{λ^*} for $\boldsymbol{x}^*$ to obtain label y^*.
  Update dataset $\mathcal{D}$ with $(\boldsymbol{x}^*, \lambda^*, y^*)$.
  Re-Train surrogate with $\mathcal{D}$.
end while

a high-fidelity λ_1, instead of querying g_{λ_1} directly, we can save some money by querying a cheaper source g_{λ_2} to obtain the label $g_{\lambda_2}(\boldsymbol{x})$. If, in addition, we have some understanding on the correlation between g_{λ_1} and g_{λ_2}, then $g_{\lambda_1}(\boldsymbol{x})$ can be inferred from $g_{\lambda_2}(\boldsymbol{x})$.

We propose a novel learning framework with multi-fidelity sources $\mathcal{G} = \{g_\lambda\}_{\lambda\in\Lambda}$. The sources are approximated with a single surrogate function $f : \mathcal{X} \times \Lambda \mapsto \mathcal{Y}$, where $f(\cdot, \lambda)$ estimates $g_\lambda(\cdot)$ for each fidelity index $\lambda \in \Lambda$. The direct output of f is a logits vector $\eta = f(\boldsymbol{x}, \lambda|\theta)$ of size C and the predicted label is the index of the largest entry of η. The purpose of this design is to allow the model to learn the correlations in both instance space $\mathcal{X}$ and fidelity space Λ. Conceptually, we have just extended the input dimension by one. However, in practice, simply feeding the concatenation of the instance vector and the fidelity index (or its one-hot encoding) into a model is not feasible, as discrete representations are not informative enough to describe the complex relationships between fidelities. To address this problem, we transform the fidelity indices into embeddings, which are vectors of the instance dimension. The instance and embedding are concatenated along an extra channel dimension before feeding into the model. This idea is originally proposed to describe the multiple degrees of similarity among words [90], which are discrete in nature.

In the multi-fidelity active learning, we need to select the instance to query as we do in active learning, and also a source in $\mathcal{G}$. The acquisition function in the multi-fidelity setting is defined to be a function that maps both instance and fidelity level to a score, *i.e.*, $\mathcal{A} : \mathcal{X} \times \Lambda \mapsto \mathbb{R}^+$. At every iteration, we find the maximizer $(\boldsymbol{x}^*, \lambda^*)$ of $\mathcal{A}$ in the product space $\mathcal{X} \times \Lambda$, and query the source g_{λ^*} indexed by λ^* for the instance $\boldsymbol{x}^*$.

So far, we have not taken consideration of the costs associated with the sources, which should play a role in our algorithm to prevent us from querying the expensive high-fidelity source recklessly. One straight forward approach is to introduce a heuristic regularizer $\mathcal{R}$

that balances between the score and the cost, *i.e.*$\widetilde{\mathcal{A}}(\boldsymbol{x}^*, \lambda) = \mathcal{R}\left(\mathcal{A}(\boldsymbol{x}^*, \lambda), \text{cost}(\lambda^*)\right)$, where $\mathcal{A}$ is some information measure, and $\widetilde{\mathcal{A}}$ realizes the concept of the information gain per cost [72, 131, 83, 31, 114]. However, the choice of the regularizer in this approach is highly problem-dependent and may require a lot of engineering work, *e.g.*, the cost needs to be scaled carefully to balance the scores from $\mathcal{A}$.

We propose a strategy with the ease of no hyper-parameter tuning. Assume the user has a planned the number of queries N_λ to each individual source g_λ, and define $\{N_i\}_{i=0}^M$ to be the query budgets. At each iteration, given the current statistics of queries $\{n_i\}_{i=0}^M$, the remaining queries are $\{N_i - n_i\}_{i=0}^M$. We sample m query candidates in the form $\{(\boldsymbol{x}^{(i)}, \lambda^{(i)})\}_{i=0}^m$, where, for each i, $\boldsymbol{x}^{(i)}$ is sampled from $\mathcal{X}$ randomly, and $\lambda^{(i)}$ is sampled from the fidelity space $\Lambda = \{i\}_{i=0}^M$ under the scaled probabilities $\left\{(N_i - n_i)/(\sum_{j=0}^M N_j - n_j)\right\}_{i=0}^M$. In particular, if the query budget for the source g_i is exhausted, *i.e.*, $N_i = n_i$, the fidelity index i is sampled with zero probability, and the algorithm will not query g_i anymore. The information of the cost is implied by the query budget from the user: the cheaper a source is, the larger query budget that user might assign to this source. The overall MFAL procedure is outlined in Algorithm 5.

7.3 Experiments

Throughout the experiments in this section, we evaluate the surrogate at fidelity 0, *i.e.*, $f(\cdot, 0)$, as an approximation of the HF source g_0. However, the inference scheme is flexible: in the applications where the credibility of the information sources is unknown, one can predict with the weighted average score from $f(\cdot, \lambda)$, for all $\lambda \in \Lambda$. The timing benchmarks are evaluated on a single core of an 8-core processor, Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz, with 64 GB of memory.

7.3.1 Synthetic Data

Our experiments on synthetic data consider the setting where one HF source and one or two LF sources are available. The instance domain $\mathcal{X}$ is a hypercube and the direct output of the source is some real number. We define $\boldsymbol{x} \in \mathcal{X}$ to be of class c if $g_0(\boldsymbol{x}) \in [a_{c-1}, a_c]$, for some pre-defined threshold values $-\infty = a_0 < a_1 < a_2 < \ldots < a_C = \infty$. For evaluation, we consider the benchmarks **Trig**-2, **De**-2, **Friedman**-5 and **CNN**-d, where the dashed number indicates the dimension of the input. See Section 7.3.3 for more details.

A multilayer perceptron (MLP) is trained to fit the sampled data with Adam optimizer [68] for 400 epochs at max. For multi-fidelity models, we use an additional embed-

ding layer [90] to transform the fidelity index into an embedding vector that matches the dimension of the instance. The input is obtained by concatenating the instance and fidelity representation along the channel axis. Our model architecture is designed as follows:

$\texttt{Input} - \texttt{fc}(\textit{Dim}, 500) - \texttt{ReLU} - \texttt{fc}(500, 200) - \texttt{ReLU} - \texttt{ReShape} - \texttt{fc}(200 \times \textit{NumChl}, \textit{NumCls})$.

We also conduct sanity checks on the expressiveness of our model and the complexity of the benchmarks in Section 7.3.3. Our result shows that the model can learn from the sampled data effectively. Therefore, its accuracy on testing data is a legit measure to evaluate the data sampling algorithms.

7.3.1.1 Active Learning with Multi-Fidelity Sources

The main results for synthetic data benchmarks are presented in Figure 7.2, where we set the total number of HF queries to be $100, 200, 2k, 10k$ for De-2, Friedman-5, CNN-20, CNN-100, respectively, and set the total number of queries from each LF source to be the same. Four sampling algorithms are considered, which are Latin Hypercube random sampling (RD), Active Learning sampling (AL), Multi-Fidelity Latin Hypercube random sampling (MFRD), and Multi-Fidelity Active Learning sampling (MFAL). We compare the performance of these algorithms by plotting the testing accuracy of our model over the number of HF queries. Note that in a multi-fidelity experiment, performance is only recorded when the HF source is queried.

The advantage of active learning is clear in lower-dimensional problems but is diminishing as the dimension grows [134, 63]. However, data from less accurate and cheaper information sources improves the performance of the surrogate by a noticeable margin, in comparison against both the LF sources and the surrogate trained with the HF source only. We further illustrate the sampling dynamics of MFAL in Section 7.3.4.

7.3.1.2 Compare with Gaussian Process

Gaussian process is a popular approach for uncertainty estimation, and is used to build the acquisition function in active learning. However, for classification tasks, the Gaussian likelihood assumption on the data is not appropriate. In particular, one must resort to approximate inference techniques, such as the widely used Laplace approximation and the expectation propagation algorithm. Markov-chain Monte Carlo (MCMC) methods provide a powerful alternative to approximate posterior inference, but standard sampling schemes such as Gibbs and Metropolis-Hastings suffer from slow convergence rates due to strong correlations in the Gaussian process posterior.

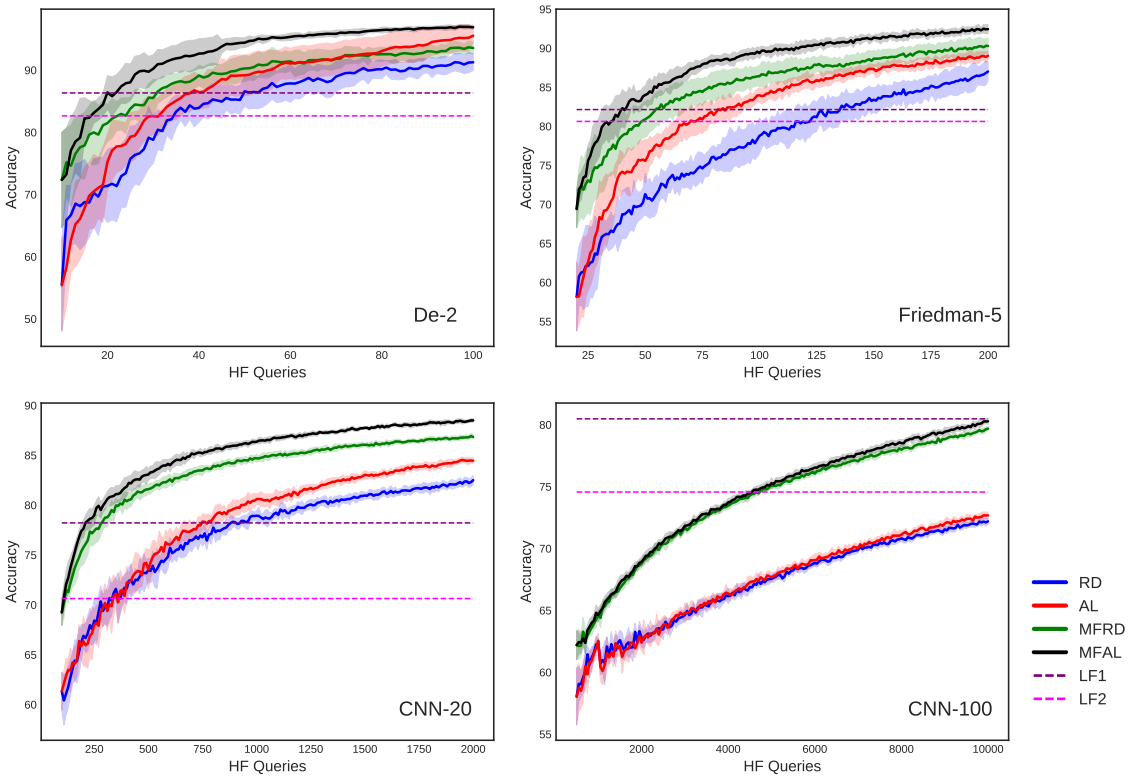


Figure 7.2: Accuracy over the number of HF Queries. The credibilities of the LF sources are indicated by the horizontal dotted lines. The query budgets for the benchmarks are {100,100,100}, {200,200,200}, {2k,2k,2k}, {10k,10k,10k}, for De-2, Friedman-5, CNN-20, CNN-100, respectively. Results are computed from 10 trials. Active learning exhibits advantages in the low-dimensional problems, whereas multi-fidelity data is increasingly helpful in the high-dimensional problems. Our MFAL algorithm out-performs the baselines by a noticeable margin.

In Table 7.1, we compare our algorithm with the sparse version of Gaussian Process Multi-Fidelity Active Learning (sGPMFAL) algorithm proposed in [109], on **Trig**-2 benchmark. We initialize with 10 HF instances and 45 LF instances (30 inducing points) by Latin Hypercube random sampling and run the algorithm until 34 HF instances are sampled in total. For Gaussian process, the first 1k samples are used to tune the step size of the sampler, and the parameters are estimated by the subsequent 1k samples, collected with one chain; the target accept probability is set to be 0.95. The active learning samplings are chosen from 1k candidates, and the inference is made with 100 posterior predictive samples. The code is open-sourced by Sahli Costabal et al. [109], where the sampler is implemented in PyMC3 [113].

Our experiments in Table 7.1 show that MLP performs on-par with GP. However, our algorithm is faster since the inference from MLP only requires a single forward pass, instead of the expensive MCMC sampling. Besides, GP suffers from the cubic and linear complexities with respect to the number of training data and output classes, respectively. On

Method	Query Budget	Train Time (s)	Sample Time (s)	Inference Time (s)	Test Acc (%)
GPAL	$\{34\}$	711	904	39	91.19
sGPMFAL	$\{34, 45\}$	12,594	2,007	94	97.26
AL	$\{34\}$	969	27	5×10^{-2}	94.2
MFAL	$\{34, 45\}$	1,251	34	5×10^{-2}	96.83

Table 7.1: Performance comparison between our MLP training and GP training. The test accuracy is averaged from 5 trials. The training and inference of MLP are faster, with competitive performance.

the other hand, the time required for MLP training admits linear scaling to the number of training data and can be easily adapted to any input dimension and number of classes, upon proper modification of the input header layer and classification layer of the model.

7.3.2 Real Data - 2D Cahn-Hilliard for Spinodal Decomposition of A-B Binary Alloy

We apply our algorithm to a two-dimensional phase-field simulation of the spinodal decomposition using Cahn-Hilliard equation [23]. This simulation describes the spinodal decomposition in an A-B binary alloy: if a high-temperature mixture of two metallic components is rapidly cooled to a lower temperature, then the mixture starts to unmix from one thermodynamic phase to two coexisting phases. Visualizations of the simulation are provided in Section 7.4.

Consider a spinodal decomposition in a virtual A-B alloy governed by a two-dimensional phase-field simulation of the spinodal decomposition using Cahn-Hilliard equation. The temporal evolution of the concentration c of the B atom is derived from the Cahn-Hilliard equation given as

$$\frac{\partial c}{\partial t} = \nabla \cdot (M_c \nabla \mu) \,. \tag{7.5}$$

Here, μ is the diffusion potential of B atom

$$\mu = \frac{\delta G}{\delta c} = RT \left[\log(c) - \log(1-c)\right] + L(1-2c) - a_c \nabla^2 c, \tag{7.6}$$

with the total free energy G of a system, gas constant R, atom interaction constant L, and

Method	Query Budget	Query Time (s)	Train Time (s)	Sample Time (s)	Test Acc (%)
Exp1: Early Termination					
AL	{25}	126,366	2,410	101	84.82
MFAL	{20,20}	116,711	2,521	177	95.31
Exp2: Coarse Discretization					
AL	{25}	135,490	1,849	90	83.61
MFAL	{20,20}	127,366	2,228	162	93.77

Table 7.2: Performance comparison between our AL and MFAL algorithms. The accuracies of the LF sources are 79.58% and 82.32% for Exp1 and Exp2, respectively. The test accuracy is averaged from 3 trials. To demonstrate the advantage of MFAL, the query budgets are selected in a way that MFAL takes less query time. With the help of LF queries, MFAL out-performs AL by a significant margin.

temperature T. And the diffusion mobility M_c of B atom is given by

$$M_c = \frac{D_A}{RT}\left[c + \frac{D_B}{D_A}(1-c)\right] c(1-c), \tag{7.7}$$

where D_A and D_B are the diffusion coefficients of A and B atoms, respectively.

The numerical simulation[1] is performed on the square domain of size 60×60 nm^2 with periodic boundary conditions. The first order Euler method is used for time-integration and the second order central finite difference method is used for spatial derivatives. Influential factors for this diffusion process are the absolute initial temperature T and the diffusion coefficients D_A,D_B of the A, B atoms, respectively. We fix the diffusion coefficient of the A atom to be $D_A = 0.00015$ m^2/s and consider a parameter domain of $(T, D_B) \in [550°, 700°] \times [0.00001\text{m}^2/\text{s}, 0.0001\text{m}^2/\text{s}]$. The objective is to model the phase of the B atom at the midpoint of the spatial domain in the long run, which is categorized into three classes by the concentration thresholds $\frac{1}{3}, \frac{2}{3}$.

7.3.3 Benchmarks

We evaluate our algorithm on several benchmarks defined as follows:

In **Trig**-2 Benchmark [109], the sources are functions of the form $g_\lambda(x_1, x_2) = \boldsymbol{\beta}_0^\lambda + \sin(\boldsymbol{\beta}_1^\lambda \pi x_1)/\boldsymbol{\beta}_2^\lambda - x_2$, for $x_i \in [-\pi, \pi], i \in \{1, 2\}$. The source parameters $\boldsymbol{\beta}^\lambda$ for fidelities $0, 1$ are defined as $\boldsymbol{\beta}^0 = (0.5, 2.5, 3)^T$, $\boldsymbol{\beta}^1 = (0.45, 2.2, 2.5)^T$. The intermediate threshold value is 0 for two classes. The accuracy of g_1 is 87.09%.

[1] http://web.tuat.ac.jp/~yamanaka/pcoms2019/Cahn-Hilliard-2d.html

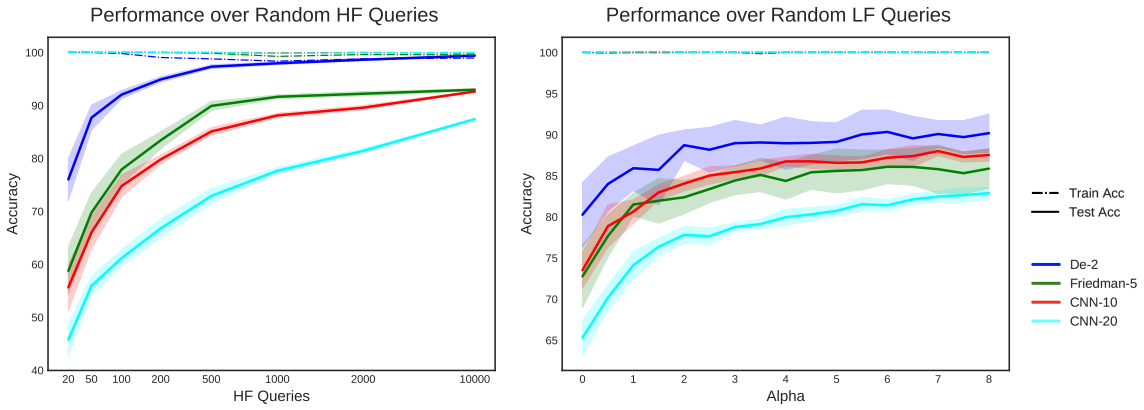


Figure 7.3: (L) Performance of 500-200 MLP over the number of HF queries. The model is trained with Adam for 400 epochs. The high training accuracy implies the model has sufficient capacity to fit the training data; testing accuracy increases as more training data is available. (R) Performance over MF query budgets of the form $\{N, \alpha N, \alpha N\}$, where N is 25,50,100,200 for De-2, Friedman-5, CNN-10, CNN-20, respectively. Testing accuracy increases as more LF data is available.

In **De**-2 Benchmark, the sources are functions of the form $g_\lambda(x_1, x_2) = \boldsymbol{\beta}_0^\lambda \sin(x_1) + \boldsymbol{\beta}_1^\lambda x_2^4 \sin(x_1)$, for $x_i \in [-\pi, \pi], i \in \{1, 2\}$. The source parameters $\boldsymbol{\beta}^\lambda$ for fidelities $0, 1, 2$ are defined as $\boldsymbol{\beta}^0 = (1, 0.1)^T$, $\boldsymbol{\beta}^1 = (1.3, 0.07)^T$, $\boldsymbol{\beta}^2 = (0.7, 0.13)^T$. The intermediate threshold values are ± 0.80 for three classes. The accuracy of g_1, g_2 is $86.08\%, 82.60\%$, respectively.

In **Friedman**-5 Benchmark, the sources are functions of the form $g_0(x_1, x_2, x_3, x_4, x_5) = \boldsymbol{\beta}_0^\lambda \sin(\pi x_1 x_2) + \boldsymbol{\beta}_1^\lambda (x_3 - \boldsymbol{\beta}_2^\lambda)^2 + \boldsymbol{\beta}_3^\lambda x_4 + \boldsymbol{\beta}_4^\lambda x_5$, for $x_i \in [0, 1], i \in \{1, 2, 3, 4, 5\}$. The source parameters $\boldsymbol{\beta}^\lambda$ for fidelities $0, 1, 2$ are defined as $\boldsymbol{\beta}^0 = (0.4, 0.8, 0.5, 0.4, 0.2)^T$, $\boldsymbol{\beta}^1 = (0.48, 0.75, 0.55, 0.35, 0.25)^T$, $\boldsymbol{\beta}^2 = (0.32, 0.85, 0.45, 0.45, 0.15)^T$. The intermediate threshold values are $0.50, 0.66$ for three classes. The accuracy of g_1, g_2 is $81.35\%, 79.91\%$, respectively.

In **CNN**-d Benchmark, the HF source is built with a randomly initialized deep convolutional neural network that takes inputs in the hypercube $\mathcal{X} = [0, 1]^d$ and outputs one of the three classes. The LF sources are obtained by perturbing the weights of the HF source.

The training and testing accuracy of the MLP defined in Section 7.3.1 on random HF queries are presented in Figure 7.3(L). Our chosen architecture has enough capacity to fit up to 10k training data with more than 95% accuracy. This indicates that the model can learn from the data efficiently, and its accuracy on testing data is a legit measure to evaluate data sampling algorithms. For the benchmark of a higher dimension, the model requires more data to achieve a certain level of testing accuracy. This is expected as high-dimensional learning is a challenging task.

In Figure 7.3(R), we fix the number of HF queries to be N and consider query budgets

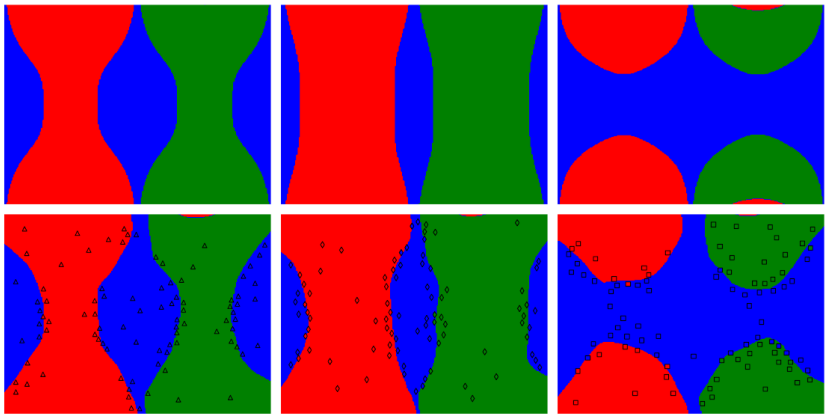


Figure 7.4: Sampling dynamics of MFAL. The ground truths for HF, LF1, LF2 are in the first row from left to right and the corresponding predicted contour plots are in the second row. MFAL focuses on sampling the "bottleneck" region at the center, where the disagreement among the sources arises.

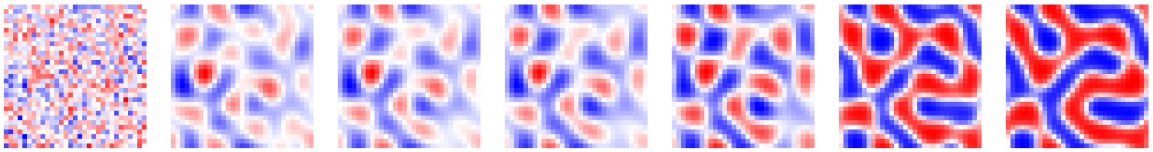
of the form $\{N, \alpha N, \alpha N\}$, with some multiplier $\alpha \geq 0$. Multi-fidelity data is randomly sampled according to the query budget and the accuracy is reported for different αs, ranging from 0 to 8. In particular, $\alpha = 0$ implies learning with HF source only. In this experiment, N is chosen to be 25, 50, 100, 200 for benchmarks **De**-2, **Friedman**-5, **CNN**-10, **CNN**-20, respectively.

When extra information sources are available, we can query the noisy but cheaper sources to obtain a multi-fidelity dataset that improves the surrogate's performance. For example, in Figure 7.3(L), it takes 100 random HF queries to achieve about 78% accuracy on the Friedman-5 benchmark. However, as shown in Figure 7.3(R), we get a model with 85% accuracy if we make 50 HF queries and 400 queries for each of the LF sources. More LF queries often improve the performance of the surrogate, as they help to learn from the LF sources and their correlations with the HF source.

7.3.4 Learning the Region of Disagreement

In Figure 7.4, we present the **De**-2 example to illustrate how data is sampled and its impact on the surrogate training. The decision regions of the ground truths (GT) and surrogate trained with MFAL sampled data are shown in the first and second row respectively, where the columns from left to right are the associated fidelities: HF, LF1, LF2. Note that for the surrogate f, we compute $f(\mathcal{X}, 0), f(\mathcal{X}, 1), f(\mathcal{X}, 2)$ over a discretized grid of $\mathcal{X}$ to estimate the decision boundary at each fidelity level. The queried points sampled at each fidelity are also shown in the graph.

In classical active learning, the points sampled tend to be near the decision boundary of the ground truth, where the prediction uncertainty arises. In multi-fidelity learning, one



1 steps 1k steps 2k steps 3k steps 5k steps 7k steps 9k steps

Figure 7.5: Visualizations of 2D Cahn-Hilliard simulation. Blue and red indicate high concentrations of A atom and B atom, respectively.

(a) Exp1 HF (b) Exp1 LF (c) Exp2 HF (d) Exp2 LF

Figure 7.6: The decision regions of the ground truths on solutions at the midpoint of the domain from 2D Cahn-Hilliard (CH) simulation parametrized by a temperature coefficient (horizontal axis) and a diffusion coefficient (vertical axis). The solution is categorized into three classes by the concentration thresholds $1/3, 2/3$. (**ab**) Ground Truths for Exp1. The query times of HF **(a)** and LF **(b)** are 5038 and 543 seconds, respectively. The accuracy of LF is 75.98%. (**cd**) Ground Truths for Exp2. The query times of HF **(c)** and LF **(d)** are 5970 and 595 seconds, respectively. The accuracy of LF is 82.32%.

makes a prediction based on the noisy information from the LF sources, therefore, it's important to judge how reliable LF data is for HF prediction. MFAL focuses on the regions of disagreements among the information sources, *e.g.*, the "bottleneck" region at the center. Surrogate trained with data sampled from MFAL is less likely to be distracted from the incorrect LF labels, as it's made aware of the difference among the sources.

7.4 2D Cahn-Hilliard for Spinodal Decomposition of A-B Binary Alloy

Two experiments are conducted in Table 7.2. In the first experiment, HF and LF sources are the solutions at 20k and 2k time steps of the simulation at the midpoint of the domain discretized by 31×31 grids, respectively. The time step sizes are adjusted to ensure the convergence; HF simulation takes 5038 seconds and LF simulation takes 543 seconds. In the second experiment, HF and LF sources are the solutions at 10k and 2.5k time steps of the simulation at the midpoint of the domain discretized by 41×41 grids and 27×27 grids, respectively. HF simulation takes 5970 seconds and LF simulation takes 595 seconds. Visualizations of the ground truths are provided in Section 7.4. To demonstrate the advantage

of MFAL over AL, we design query budgets so that the overall query time for MFAL is lower. The training and sampling time of our algorithm is negligible in comparison with simulation time, and MFAL out-performs AL with less query time. The benefits of MFAL become more significant as the input dimension or the domain size grows, where the HF query time scales up.

7.5 Conclusion

We construct a Multi-Fidelity learning framework with neural networks for surrogate training on classification tasks. Our training is faster than Gaussian process training with on-par performance. Also, we propose a Multi-Fidelity Active Learning algorithm to further improve the efficiency of the sampled data under a limited budget. The proposed methodology significantly out-performs Active Learning and Multi-Fidelity Random sampling.

Bibliography

[1] S. Aaronson. Why quantum chemistry is hard. *Nature Physics*, 5(10):707–708, 2009.

[2] P.-A. Absil, C. G. Baker, and K. A. Gallivan. Trust-region methods on Riemannian manifolds. *Foundations of Computational Mathematics*, 7(3):303–330, 2007.

[3] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[4] H. Alghassi, A. Deshmukh, N. Ibrahim, N. Robles, S. Woerner, and C. Zoufal. A variational quantum algorithm for the feynman-kac formula. *arXiv preprint arXiv:2108.10846*, 2021.

[5] S.-I. Amari. Natural gradient works efficiently in learning. *Neural computation*, 10 (2):251–276, 1998.

[6] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, 2016.

[7] S. Barison, F. Vicentini, and G. Carleo. An efficient quantum algorithm for the time evolution of parameterized circuits. *Quantum*, 5:512, July 2021. ISSN 2521-327X. doi: 10.22331/q-2021-07-28-512. URL https://doi.org/10.22331/q-2021-07-28-512.

[8] T. D. Barrett, A. Malyshev, and A. Lvovsky. Autoregressive neural-network wavefunctions for ab initio quantum chemistry. *Nature Machine Intelligence*, 4(4):351–358, 2022.

[9] R. J. Bartlett and M. Musiał. Coupled-cluster theory in quantum chemistry. *Reviews of Modern Physics*, 79(1):291, 2007.

[10] A. Basudhar, S. Missoum, and A. H. Sanchez. Limit state function identification using support vector machines for discontinuous responses and disjoint failure domains. *Probabilistic Engineering Mechanics*, 23(1):1–11, 2008.

[11] J. Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.

[12] C. Beck, A. Jentzen, et al. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *Journal of Nonlinear Science*, 29(4):1563–1619, 2019.

[13] Y. Bengio and S. Bengio. Modeling high-dimensional discrete data with multi-layer neural networks. In *Advances in Neural Information Processing Systems*, 2000.

[14] B. J. Bichon, M. S. Eldred, L. P. Swiler, S. Mahadevan, and J. M. McFarland. Efficient global reliability analysis for nonlinear implicit performance functions. *AIAA Journal*, 46(10):2459–2468, 2008.

[15] M. Born and R. Oppenheimer. Zur quantentheorie der molekeln. *Annalen der physik*, 389(20):457–484, 1927.

[16] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a matlab toolbox for optimization on manifolds. *The Journal of Machine Learning Research*, 15(1): 1455–1459, 2014.

[17] N. Boumal, V. Voroninski, and A. S. Bandeira. The non-convex burer–monteiro approach works on smooth semidefinite programs. In *Advances in Neural Information Processing Systems*, 2016.

[18] S. Bravyi, D. Gosset, R. König, and K. Temme. Approximation algorithms for quantum many-body problems. *Journal of Mathematical Physics*, 60(3):032203, 2019.

[19] S. B. Bravyi and A. Y. Kitaev. Fermionic quantum computation. *Annals of Physics*, 298(1):210–226, 2002.

[20] J. Bruna, B. Peherstorfer, and E. Vanden-Eijnden. Neural Galerkin scheme with active learning for high-dimensional evolution equations. *arXiv preprint arXiv:2203.01360*, 2022.

[21] S. L. Brunton, B. R. Noack, and P. Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52(1):477–508, 2020.

[22] S. Burer and R. D. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming (series B*, 95:2003, 2001.

[23] J. W. Cahn and J. E. Hilliard. Free energy of a non-uniform system in interfacial energy. *The Journal of Chemical Physics*, 28(2):258–267, 1958.

[24] G. Carleo and M. Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

[25] G. Carleo, K. Choo, D. Hofmann, J. E. Smith, T. Westerhout, F. Alet, E. J. Davis, S. Efthymiou, I. Glasser, S.-H. Lin, and et al. Netket: A machine learning toolkit for many-body quantum systems. *SoftwareX*, 10:100311, 2019.

[26] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[27] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.

[28] K. Choo, A. Mezzacapo, and G. Carleo. Fermionic neural-network states for ab-initio electronic structure. *Nature communications*, 11(1):1–7, 2020.

[29] F. Coester and H. Kümmel. Short-range correlations in nuclear wave functions. *Nuclear Physics*, 17:477–485, 1960.

[30] S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[31] P. Donmez and J. G. Carbonell. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of the 17th ACM conference on Information and knowledge management*, 2008.

[32] C. Dribusch, S. Missoum, and P. Beran. A multifidelity approach for the construction of explicit decision boundaries: application to aeroelasticity. *Structural and Multidisciplinary Optimization*, 42(5):693–705, 2010.

[33] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[34] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542:115–118, 2017.

[35] A. Fallah, A. Mokhtari, and A. Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 1082–1092, 2020.

[36] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[37] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[38] F. Fontanela, A. Jacquier, and M. Oumgari. A quantum algorithm for linear PDEs arising in finance. *SIAM Journal on Financial Mathematics*, 12(4):SC98–SC114, 2021.

[39] A. I. Forrester and A. J. Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1):50–79, 2009.

[40] A. I. Forrester, A. Sóbester, and A. J. Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 463(2088):3251–3269, 2007.

[41] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 2016.

[42] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, 6(6):721–741, 1984.

[43] M. Germain, K. Gregor, I. Murray, and H. Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, 2015.

[44] M. Germain, K. Gregor, I. Murray, and H. Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, 2015.

[45] M. B. Giles. Multilevel Monte Carlo Path Simulation. *Operations Research*, 56(3): 607–617, 2008.

[46] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[47] J. Gomes, K. A. McKiernan, P. Eastman, and V. S. Pande. Classical quantum optimization with neural network quantum states. *arXiv preprint arXiv:1910.10675*, 2019.

[48] A. A. Gorodetsky, G. Geraci, M. S. Eldred, and J. D. Jakeman. A generalized approximate control variate framework for multifidelity uncertainty quantification. *Journal of Computational Physics*, 408:109257, 2020.

[49] P. J. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[50] T. Guillaume. On the multidimensional black–scholes partial differential equation. *Annals of Operations Research*, 281(1):229–251, 2019.

[51] I. L. Gutiérrez and C. B. Mendl. Real time evolution with neural-network quantum states. *Quantum*, 6:627, 2022.

[52] J. Haegeman, J. I. Cirac, T. J. Osborne, I. Pižorn, H. Verschelde, and F. Verstraete. Time-dependent variational principle for quantum lattices. *Physical review letters*, 107(7):070601, 2011.

[53] A. L. Haji-Ali, F. Nobile, L. Tamellini, and R. Tempone. Multi-index stochastic collocation for random pdes. *Computer Methods in Applied Mechanics and Engineering*, 306:95–122, 2016.

[54] A. L. Haji-Ali, F. Nobile, and R. Tempone. Multi-index Monte Carlo: when sparsity meets sampling. *Numerische Mathematik*, 132(4):767–806, 2016.

[55] B. L. Hammond, W. A. Lester, and P. J. Reynolds. *Monte Carlo methods in ab initio quantum chemistry*, volume 1. World Scientific, 1994.

[56] J. Han, A. Jentzen, et al. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.

[57] J. Han, A. Jentzen, and E. Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[58] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, G. H. Tison, C. Bourn, M. P. Turakhia, and A. Y. Ng. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature Medicine*, 25(3):530–530, 2019.

[59] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[60] M. Hibat-Allah, M. Ganahl, L. E. Hayward, R. G. Melko, and J. Carrasquilla. Recurrent neural network wave functions. *Physical Review Research*, 2(2):023358, 2020.

[61] M. D. Hoffman and A. Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1): 1593–1623, 2014.

[62] C. Huré, H. Pham, and X. Warin. Deep backward schemes for high-dimensional nonlinear pdes. *Mathematics of Computation*, 89(324):1547–1579, 2020.

[63] R. Islam. Active learning for high dimensional inputs using bayesian convolutional neural networks. Master's thesis, 2016.

[64] P. Jordan and E. P. Wigner. über das paulische äquivalenzverbot. In *The Collected Works of Eugene Paul Wigner*, pages 109–129. 1993.

[65] M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, 20(5): 2327–2351, 2010.

[66] S. M. Kakade. A natural policy gradient. *Advances in Neural Information Processing Systems*, 2001.

[67] M. C. Kennedy and A. O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.

[68] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[69] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improving variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, 2016.

[70] I. Kobyzev, S. Prince, and M. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[71] B. Koczor and S. C. Benjamin. Quantum natural gradient generalised to non-unitary circuits. *arXiv preprint arXiv:1912.08660*, 2019.

[72] R. Lam, D. L. Allaire, and K. E. Willcox. Multifidelity optimization using statistical surrogate modeling for non-hierarchical information sources. In *AIAA*, 2015.

[73] R. Lam, K. Willcox, and D. H. Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. In *Advances in Neural Information Processing Systems*, 2016.

[74] S. Langhoff. *Quantum mechanical electronic structure calculations with chemical accuracy*, volume 13. Springer Science & Business Media, 2012.

[75] H. Larochelle and I. Murray. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 29–37, 2011.

[76] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *SIGIR*, 1994.

[77] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[78] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2017.

[79] D. Luo and B. K. Clark. Backflow transformations via neural networks for quantum many-body wave functions. *Physical review letters*, 122(22):226401, 2019.

[80] L. Malagò, M. Matteucci, and B. Dal Seno. An information geometry perspective on estimation of distribution algorithms: boundary analysis. In *Proceedings of the 10th annual conference companion on Genetic and evolutionary computation*, pages 2081–2088, 2008.

[81] L. Malagò, M. Matteucci, and G. Pistone. Towards the geometry of estimation of distribution algorithms based on the exponential family. In *Proceedings of the 11th workshop proceedings on Foundations of genetic algorithms*, pages 230–242, 2011.

[82] G. R. Marple, D. Gorsich, P. Jayakumar, and S. Veerapaneni. An active learning framework for constructing high-fidelity mobility maps. *IEEE Transactions on Vehicular Technology*, 70(10):9803–9813, 2021.

[83] A. N. Marques, R. R. Lam, and K. E. Willcox. Contour location via entropy reduction leveraging multiple information sources. In *Advances in Neural Information Processing Systems*, 2018.

[84] A. N. Marques, R. Lam, A. Chaudhuri, M. M. Opgenoord, and K. E. Willcox. A multifidelity method for locating aeroelastic flutter boundaries. In *AIAA*, 2019.

[85] J. R. McClean, N. C. Rubin, K. J. Sung, I. D. Kivlichan, X. Bonet-Monroig, Y. Cao, C. Dai, E. S. Fried, C. Gidney, B. Gimby, et al. Openfermion: the electronic structure package for quantum computers. *Quantum Science and Technology*, 5(3):034014, 2020.

[86] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[87] A. McLachlan. A variational solution of the time-dependent schrodinger equation. *Molecular Physics*, 8(1):39–44, 1964.

[88] W. L. McMillan. Ground state of liquid he 4. *Physical Review*, 138(2A):A442, 1965.

[89] D. Mechergui and P. Jayakumar. Efficient generation of accurate mobility maps using machine learning algorithms. *Journal of Terramechanics*, 88:53–63, 2020.

[90] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *ICLR Workshops Track*, 2013.

[91] T. Misawa, S. Morita, K. Yoshimi, M. Kawamura, Y. Motoyama, K. Ido, T. Ohgoe, M. Imada, and T. Kato. mvmc—open-source software for many-variable variational monte carlo method. *Computer Physics Communications*, 235:447–462, 2019.

[92] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, USA, 2005. ISBN 0521835402.

[93] G. F. Montúfar, J. Rauh, and N. Ay. Expressive power and approximation errors of restricted boltzmann machines. In *Advances in Neural Information Processing Systems*, 2011.

[94] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[95] Y. Nomura, A. S. Darmawan, Y. Yamaji, and M. Imada. Restricted boltzmann machine learning for solving strongly correlated quantum systems. *Physical Review B*, 96(20):205152, 2017.

[96] Y. Ollivier, L. Arnold, A. Auger, and N. Hansen. Information-geometric optimization algorithms: A unifying picture via invariance principles. *The Journal of Machine Learning Research*, 18(1):564–628, 2017.

[97] E. J. Parish and K. T. Carlberg. Time-series machine-learning error models for approximate solutions to parameterized dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 365:112990, 2020.

[98] P. Perdikaris, D. Venturi, J. O. Royset, and G. E. Karniadakis. Multi-fidelity modelling via recursive co-kriging and gaussian–markov random fields. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2179): 20150018, 2015.

[99] P. Perdikaris, D. Venturi, and G. E. Karniadakis. Multifidelity information fusion algorithms for high-dimensional systems and massive data sets. *SIAM Journal on Scientific Computing*, 38(4):B521–B538, 2016.

[100] D. Pfau, J. S. Spencer, A. G. Matthews, and W. M. C. Foulkes. Ab initio solution of the many-electron schrödinger equation with deep neural networks. *Physical Review Research*, 2(3):033429, 2020.

[101] M. Poloczek, J. Wang, and P. Frazier. Multi-information source optimization. In *Advances in Neural Information Processing Systems*, 2017.

[102] J. Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[103] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker. Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41(1): 1–28, 2005.

[104] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378: 686–707, 2019.

[105] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[106] I. Rechenberg. Evolutionsstrategien. In *Simulationsmethoden in der Medizin und Biologie*, pages 83–114. Springer, 1978.

[107] M. Reh and M. Gärttner. Variational monte carlo approach to partial differential equations with neural networks. *arXiv preprint arXiv:2206.01927*, 2022.

[108] F. Rendl, G. Rinaldi, and A. Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121(2):307, 2010.

[109] F. Sahli Costabal, P. Perdikaris, E. Kuhl, and D. E. Hurtado. Multi-fidelity classification using gaussian processes: Accelerating the prediction of large-scale computational models. *Computer Methods in Applied Mechanics and Engineering*, 357: 112602, 2019.

[110] F. Sahli-Costabal, K. Seo, E. Ashley, and E. Kuhl. Classifying drugs by their arrhythmogenic risk using machine learning. *Biophysical Journal*, 118(5):1165–1176, 2020.

[111] R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In *International Conference on Machine Learning*, pages 872–879, 2008.

[112] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[113] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck. Probabilistic programming in python using pymc3. *PeerJ Computer Science*, 2:e55, 2016.

[114] S. Sarkar, S. Mondal, M. Joly, M. E. Lynch, S. D. Bopardikar, R. Acharya, and P. Perdikaris. Multifidelity and multiscale bayesian framework for high-dimensional engineering design and calibration. *Journal of Mechanical Design*, 141(12):121001, 2019.

[115] H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie: mit einer vergleichenden Einführung in die Hill-Climbing-und Zufallsstrategie*, volume 1. Springer.

[116] B. Settles. Active learning literature survey. Computer sciences technical report, 2009.

[117] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.

[118] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[119] O. Sharir, Y. Levine, N. Wies, G. Carleo, and A. Shashua. Flowket: an open-source library based on tensorflow for running variational monte-carlo simulations on gpus. `https://github.com/HUJI-Deep/FlowKet`, 2020.

[120] O. Sharir, Y. Levine, N. Wies, G. Carleo, and A. Shashua. Deep autoregressive models for the efficient variational simulation of many-body quantum systems. *Physical review letters*, 124(2):020503, 2020.

[121] C. D. Sherrill and H. Schaefer III. Advances in quantum chemistry. *Advances in Quantum Chemistry*, 34:143–269, 1999.

[122] K. Shridhar, F. Laumann, and M. Liwicki. A comprehensive guide to bayesian convolutional neural network with variational inference. *arXiv preprint arXiv:1901.02731*, 2019.

[123] J. Sirignano and K. Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

[124] S. Sorella. Green function monte carlo with stochastic reconfiguration. *Physical review letters*, 80(20):4558–4561, 1998.

[125] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[126] J. Stokes and J. Terilla. Probabilistic modeling with matrix product states. *Entropy*, 21(12):1236, 2019.

[127] J. Stokes, J. Izaac, N. Killoran, and G. Carleo. Quantum natural gradient. *Quantum*, 4:269, 2020.

[128] J. Stokes, J. R. Moreno, E. A. Pnevmatikakis, and G. Carleo. Phases of two-dimensional spinless lattice fermions with first-quantized deep neural-network quantum states. *Physical Review B*, 102(20):205122, 2020.

[129] J. Stokes, B. Chen, and S. Veerapaneni. Numerical and geometrical aspects of flow-based variational quantum monte carlo. *arXiv preprint arXiv:2203.14824*, 2022.

[130] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147, 2013.

[131] K. Swersky, J. Snoek, and R. P. Adams. Multi-task bayesian optimization. In *Advances in Neural Information Processing Systems*, 2013.

[132] A. L. Teckentrup, P. Jantsch, C. G. Webster, and M. Gunzburger. A multilevel stochastic collocation method for partial differential equations with random input data. *SIAM/ASA Journal on Uncertainty Quantification*, 3:1046–1074, 2015.

[133] S. Thrun and L. Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[134] S. Tong. *Active learning: theory and applications*. PhD thesis, 2001.

[135] M. Troyer and U.-J. Wiese. Computational complexity and fundamental limitations to fermionic quantum monte carlo simulations. *Physical review letters*, 94(17):170201, 2005.

[136] A. van den Oord, N. Kalchbrenner, L. Espeholt, k. kavukcuoglu, O. Vinyals, and A. Graves. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, 2016.

[137] G. Verdon, M. Broughton, J. R. McClean, K. J. Sung, R. Babbush, Z. Jiang, H. Neven, and M. Mohseni. Learning to learn with quantum neural networks via classical neural networks. *arXiv preprint arXiv:1907.05415*, 2019.

[138] J. Wang, Z. Chen, D. Luo, Z. Zhao, V. M. Hur, and B. K. Clark. Spacetime neural network for high dimensional quantum dynamics. *arXiv preprint arXiv:2108.02200*, 2021.

[139] Y. Wang, J. Xiao, T. O. Suzek, J. Zhang, J. Wang, and S. H. Bryant. Pubchem: a public information system for analyzing bioactivities of small molecules. *Nucleic acids research*, 37:W623–W633, 2009.

[140] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

[141] M. Wilson, S. Stromswold, F. Wudarski, S. Hadfield, N. M. Tubman, and E. Rieffel. Optimizing quantum heuristics with meta-learning. *arXiv preprint arXiv:1908.03185*, 2019.

[142] D. Wu, L. Wang, and P. Zhang. Solving statistical mechanics using variational autoregressive networks. *Physical review letters*, 122(8):080602, 2019.

[143] K. Wu and J. Liu. Classification-based optimization with multi-fidelity evaluations. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 1126–1131, 2019.

[144] Y. Yang and M. Loog. A benchmark and comparison of active learning for logistic regression. *Pattern Recognition*, 83:401–415, 2018.

[145] J. Yao, M. Bukov, and L. Lin. Policy gradient based quantum approximate optimization algorithm. *arXiv preprint arXiv:2002.01068*, 2020.

[146] Y. Yu, H. Qian, and Y.-Q. Hu. Derivative-free optimization via classification. In *AAAI*, 2016.

[147] X. Yuan, S. Endo, Q. Zhao, Y. Li, and S. C. Benjamin. Theory of variational quantum simulation. *Quantum*, 3:191, 2019.

[148] M. D. Zeiler. Adadelta: An adaptive learning rate method, 2012.

[149] C. Zhang and K. Chaudhuri. Active learning from weak and strong labelers. In *Advances in Neural Information Processing Systems*, 2015.

[150] T. Zhao, G. Carleo, J. Stokes, and S. Veerapaneni. Natural evolution strategies and variational monte carlo. *Machine Learning: Science and Technology*, 2(2):02LT01, 2020.

[151] T. Zhao, J. Stokes, O. Knitter, B. Chen, and S. Veerapaneni. Meta variational Monte Carlo. *Third Workshop on Machine Learning and the Physical Sciences (Advances in Neural Information Processing Systems 2020)*, 2020.

[152] T. Zhao, S. De, B. Chen, J. Stokes, and S. Veerapaneni. Overcoming barriers to scalability in variational quantum monte carlo. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis*, 2021.

[153] X. Zhu, J. Lafferty, and R. Rosenfeld. *Semi-Supervised Learning with Graphs*. PhD thesis, 2005.