Quantum machine learning: on the design, trainability and noise-robustness of near-term algorithms

Skolik, A.

CHAPTER 2

Quantum computing

Quantum computing is a rapidly evolving field that has the potential to change the way we solve a number of complex computational problems. In recent years, there has been significant progress in the development of quantum computers, with researchers and companies around the world working to build these types of machines. In terms of current hardware implementations, there are two main approaches to quantum computing: quantum annealing and gate-based quantum computing. A quantum annealer is a special-purpose device tailored to solve combinatorial optimization problems, based on the idea to slowly evolve a system until it reaches its lowest-energy state, which represents the optimal solution to a given problem. Gate-based quantum computing, on the other hand, involves the use of quantum gates to manipulate qubits, the basic units of quantum information. This approach is more flexible and has the potential to perform a wider range of calculations than quantum annealing, but it is also more difficult to implement in practice due to the need to maintain the delicate quantum states of the qubits. In this thesis, we focus on the latter paradigm of quantum computing, and this chapter provides an introduction to the most important concepts in the gate-based formalism, and the specific type of algorithms we study in later chapters.

2.1 Gate model quantum computing

Quantum computers are devices that harness quantum mechanical effects to process information. In order to utilize these types of effects, one has to define a quantum system to perform operations on. A simple and broadly used approach to this are two-level systems called qubits, which form the building blocks of most quantum algorithms. Mathematically, a qubit can be represented as the quantum state

$|\psi\rangle \in \mathbb{C}^2$ with amplitudes α and β,

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \tag{2.1}$$

where $|\alpha|^2 + |\beta|^2 = 1$. The squared modulus of α and β give us the probability of measuring zero and one, respectively. We henceforth adopt bra-ket notation, where the *bra* $\langle\psi| = (\alpha^*, \beta^*)$ denotes the complex conjugate of the state $|\psi\rangle$ (*ket*), $\langle\psi|\psi\rangle$ denotes an inner product, and $|\psi\rangle\langle\psi|$ denotes an outer product. The vectors $|0\rangle$ and $|1\rangle$ form an orthonormal basis of the Hilbert space $\mathbb{C}^2$, and are therefore referred to as *basis states*. Technically, any two orthonormal states can be used as basis states for the vector space of the qubit, however, the two states above are the most common and are called the *computational basis*. These basis states are the states that we can observe in the classical world, while the linear combination of basis states in Equation (2.1) is called a *superposition*, where the coefficients α and β define the probabilities with which each of the basis states can be observed. Unlike classical probabilities, those coefficients are complex-valued and can therefore also be negative. This means that they can either add up or cancel each other out, and this constructive and destructive *interference* of amplitudes plays an important role in many quantum algorithms.

In order to manipulate the state of a qubit, one uses unitary operators which are referred to as *quantum gates*. A commonly used set of gates that can be used to implement arbitrary unitary transformations on a single qubit are the so-called Pauli operators,

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ \sigma_y = \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix},\ \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.2}$$

If one, for example, wants to implement a bitflip operation on a qubit, one can do this by applying the σ_x operator as follows,

$$\sigma_x\,|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |1\rangle\,. \tag{2.3}$$

Now, if we consider not just one but multiple qubits, the above can easily be extended by forming tensor products of the kets of a number of qubits. Let us take for example the state over the qubits $|\psi_A\rangle$, $|\psi_B\rangle$ and $|\psi_C\rangle$, then the complete state of this *qubit register* is

$$|\psi_{ABC}\rangle = |\psi_A\rangle \otimes |\psi_B\rangle \otimes |\psi_C\rangle\,, \tag{2.4}$$

and $|\psi_{ABC}\rangle$ is now an element of the Hilbert space $\mathbb{C}^{2^n}$ for $n = 3$ qubits. When we consider a register of multiple qubits, another important aspect of quantum algorithms can arise: *entanglement* between these qubits. Intuitively, entanglement means that the state of a quantum system can not simply be described by considering its individual parts. Formally, we call a state entangled if it is not a *separable state*. To understand the notion of a separable state, consider a bipartite quantum system on the Hilbert spaces $\mathcal{H}_A$ with basis $\{|a_i\rangle\}_{i=1}^k$ and $\mathcal{H}_B$ with basis $\{|b_j\rangle\}_{j=1}^l$, and a basis $\{|a_i\rangle \otimes |b_j\rangle\}$ for $\mathcal{H}_A \otimes \mathcal{H}_B$. Any pure state in this composite system can be written as

$$|\psi\rangle_{AB} = \sum_{i,j} c_{i,j}(|a_i\rangle \otimes |b_j\rangle). \tag{2.5}$$

If the state can be written as a simple tensor product of the two subsystems,

$$|\psi\rangle_{AB} = |\psi\rangle_A \otimes |\psi\rangle_B \,, \tag{2.6}$$

it is considered a separable state. Intuitively, this can be understood as a joint probability mass function that is the product of two independent marginals, i.e., $p(x, y) = p(x)p(y)$. However, the type of correlation that is present in non-separable states has no analog in the classical world and is therefore hard to understand intuitively. Entanglement is a crucial ingredient for many quantum algorithms, like the famous prime factorization algorithm by Shor [1].

So far we have discussed how to prepare and manipulate quantum states in the gate-model formalism. The final ingredient required to perform quantum computation is getting classical information out of the device, that is, performing a measurement of a given observable. Going back to the one-qubit example above, we have already discussed that the superposition state shown in Equation (2.1) can not be measured. Instead, we can only measure the basis state $|0\rangle$, which occurs with probability $|\alpha|^2$, and the basis state $|1\rangle$, with probability $|\beta|^2$. More formally, we define measurement observables in terms of a set of operators $\{M_m\}$ on the state space of the quantum system. In this thesis, we consider the special case where all M_m are orthogonal and Hermitian, called a *projective measurement*. The observable M has the spectral decomposition

$$M = \sum_m m P_m, \tag{2.7}$$

with P_m the projector onto the eigenspace of M with eigenvalue m. For the example of a computational basis measurement on one qubit $|\psi\rangle$ we have the measurement operators

$$P_0 = |0\rangle\langle 0|\,, P_1 = |1\rangle\langle 1|\,, \tag{2.8}$$

both with eigenvalue 1, and we get

$$\sum_m P_m^\dagger P_m = P_0^\dagger P_0 + P_1^\dagger P_1 = I, \tag{2.9}$$

where Equation (2.9) shows that this set of projectors satisfies the *completeness relation*, i.e., the set of projectors has to satisfy the condition that the probabilities of all measurement outcomes sum to one. After measuring outcome m, the state is then

$$|\psi_m\rangle = \frac{P_m\,|\psi\rangle}{\sqrt{p(m)}}, \tag{2.10}$$

where the probability $p(m)$ to measure outcome m is

$$p(m) = \langle\psi|\,P_m\,|\psi\rangle\,. \tag{2.11}$$

The above statement that the measurement outcome will be one of the eigenvalues of M and that the probability of measuring eigenvalue m is given by Equation (2.11) is also known as the *Born rule*. Once the observable is measured, the quantum state *collapses* and all information about its previous state is lost. Subsequent measurements of the resulting state will then always yield the same output.

Another common observable, which we will also use in this thesis, is the observable Z, also referred to as a *measurement in the Z basis*, with basis states $|0\rangle$ and $|1\rangle$, and eigenvalues 1 and -1, respectively,

$$Z = 1\,|0\rangle\langle 0| - 1\,|1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.12}$$

After defining the main ingredients of quantum algorithms above, like qubits, quantum gates, superposition, interference and entanglement, we can start writing down our own algorithms. The formalism most commonly used for this, and the one we also use in this work, are quantum circuits. In a quantum circuit, qubits are represented as *wires*, horizontal lines read from left to right. Quantum gates are then placed on these wires to indicate the operations performed on each of the qubits in the register. This formalism can be used to write down arbitrary

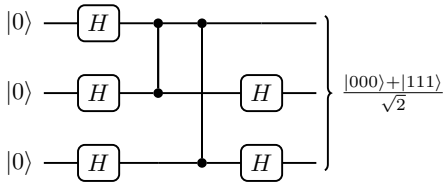

Figure 2.1: Example of a quantum circuit diagram depicting a circuit to create a Greenberger-Horne-Zeilinger (GHZ) state, an important type of entangled state. In this 3-qubit circuit, the GHZ state is prepared by applying Hadamard gates (H) and controlled phase gates (vertical lines).

quantum algorithms, and a simple example of a circuit diagram can be seen in Figure 2.1.

With the above, we are equipped to study arbitrary quantum algorithms, like the well-known prime factorization algorithm [1], or the quantum algorithm for solving linear systems of equations [5] mentioned above. In this thesis, however, we are interested in a special type of quantum algorithm tailored for near-term devices, which we will introduce in the following section.

2.2 Noisy intermediate-scale quantum computing

In Chapter 1, we already outlined the distinction between fault-tolerant algorithms, that are designed with perfect, large-scale quantum computers in mind, and so called noisy intermediate-scale quantum (NISQ) algorithms, which are more suitable for the error-prone and small quantum computers that we expect to have access to in the coming few decades. The term NISQ was coined by John Preskill in a keynote talk and accompanying article [12], and he defines these types of devices as quantum computers with around fifty to at most a few hundred qubits, which are subject to noise and which we have only imperfect control over. Apart from the number of qubits, he also emphasizes that the quality of qubits and how precisely operations can be performed on them plays a crucial role. This means that NISQ algorithms are not only restricted in the number of qubits they can use, but also in the *depth* and the total number of gates of the quantum circuits that are used. Additionally, as error correction techniques that are developed to aid

fault-tolerant quantum computation require a high overhead of additional qubits, NISQ devices are assumed to operate without error correction in order to fully utilize the small number of qubits that are available. Alongside all of the above, device-specific limitations like qubit connectivity and native gate sets have to be taken into account as well.

With these constraints, the algorithms that can successfully be run in the NISQ era are severely limited. In particular, algorithms of the type as the prime factorization and linear equation solving algorithms discussed in previous chapters [1, 5], that require the execution of a pre-defined and rigid gate sequence, are out of reach for these types of devices. However, an interesting class of algorithms specifically tailored to the limitations of these devices has emerged in the past few years: variational quantum algorithms (VQAs) [14]. VQAs are hybrid quantum-classical algorithms that outsource a large part of the heavy lifting to a classical computer. The basis for a VQA is a parametrized quantum circuit (PQC) that is run on the quantum device. Based on measurements of this quantum circuit and a given objective function, a classical optimizer computes updates of the circuit parameters. This procedure is repeated in an iterative fashion, until the circuit output matches the desired output for a given task. This approach is quite different from the one taken for designing the fault-tolerant algorithms we have discussed before. Instead of specifying a fixed algorithm in form of a quantum circuit that relies on the execution of a precise sequence of gates, in a VQA, designing this circuit is left more or less to the classical optimization routine. In general, only the structure of the parametrized gates is given in advance, and this can be tailored to the specific constraints of a certain device, like the available gate set or connections between qubits. In the following sections, we will address how the parameter optimization scheme in these types of models works, and in which areas they have shown to be promising to apply.

2.2.1 Variational quantum algorithms

A VQA consists of two components: the parametrized quantum circuit, also called *ansatz*, and the classical optimization routine that performs the parameter updates, as can be seen in Figure 2.2. In general, the structure of the ansatz can be chosen quite flexibly, and we denote a unitary parametrized by $\boldsymbol{\theta}$ as $U(\boldsymbol{\theta})$. Usually, the ansatz is structured in layers, and we write the unitary that represents a PQC of

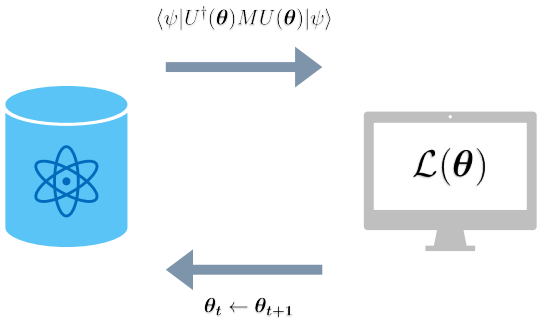

Figure 2.2: Depiction of a variational quantum-classical algorithm. The quantum computer on the left implements a quantum circuit parametrized by $\boldsymbol{\theta}$, and outputs expectation values of this circuit given an observable M. Based on this, the classical computer on the right computes a loss function $\mathcal{L}(\boldsymbol{\theta})$, and the updated parameters $\boldsymbol{\theta_{t+1}}$, which are fed into the quantum circuit for the next iteration. This process is repeated until the loss function reaches a desired value.

depth L as

$$U_L(\boldsymbol{\theta}) = \prod_{i=1}^{L} U_i(\boldsymbol{\theta}_i). \tag{2.13}$$

In the case where a PQC is used in a machine learning setting, there may also be additional parameters that serve to encode the training data into the circuit, and we will discuss these types of circuits in more detail in later chapters. For simplicity, we will omit these parameters in this chapter, as they do not influence the optimization procedure directly.

In order to optimize the parameters of the PQC given a certain task, a classical optimization routine is used. There is a wealth of options to choose from in the classical optimization literature, and indeed numerous different techniques have been explored for the optimization of PQCs [33, 34, 35, 36, 37]. One of the most popular approaches are gradient-based methods, which are the state-of-the-art optimizers used for classical neural networks. In their most basic form, called *stochastic gradient descent*, the parameters $\boldsymbol{\theta}$ of a function $f(\boldsymbol{\theta})$ are updated according to the following rule,

$$\theta_i \leftarrow \theta_i - \eta \nabla f(\theta_i), \tag{2.14}$$

where η is a step size that determines the magnitude of each update step, also called *learning rate* in the machine learning literature. There are many different versions of these types of algorithms, often with elaborate schedules to fine-tune the step size for various phases of the optimization. The Adam optimizer, for example, has additional momentum terms that tune the step size according to the steepness of the optimization landscape [38]. It depends on the given task which flavor of gradient descent is best suited, however, what they all have in common is that they require computation of partial derivatives of the function to be optimized.

2.2.1.1 Computing gradients

A simple possibility to compute gradients in PQCs is to use the *finite difference* method. With this, the gradient of an arbitrary function $f(\theta)$ can be approximated to precision $\mathcal{O}(\epsilon)$ by

$$\frac{df}{d\theta} \approx \frac{f(\theta+\epsilon) - f(\theta)}{\epsilon}. \tag{2.15}$$

With this method, the number of circuit evaluations required to compute the gradient of a PQC scales linearly with the number of parameters used in the circuit. However, this comes at the cost of only obtaining an approximation of the gradient, as well as introducing another hyperparameter, namely ϵ, to the optimization routine. Furthermore, in the case of noisy quantum hardware with imprecise control, and circuit evaluations based on a limited number of measurements, there are additional limitations on how small ϵ can be chosen in this setting.

This lack of precision in computing gradients will increase their variance during training and can therefore negatively impact the optimization routine. To alleviate the issue of only computing an ϵ-approximation of the gradients, one can also compute exact gradients for PQCs by using the so-called *parameter-shift rule* [39, 33, 40]. Consider a parametrized gate $U_G(\theta_i) = e^{-ia\theta_i G}$ with generator G that is a Hermitian linear operator, trainable parameter θ_i and a real constant a, that acts within an arbitrary unitary $U(\boldsymbol{\theta})$. If G has at most two distinct eigenvalues e_0 and e_1, the partial derivative of $\langle U(\boldsymbol{\theta})\rangle$ w.r.t. θ_i can be written as

$$\frac{d}{d\theta_i}\langle U(\boldsymbol{\theta})\rangle = r\left(\langle U(\boldsymbol{\theta}+\boldsymbol{\Delta\theta_i})\rangle - \langle U(\boldsymbol{\theta}-\boldsymbol{\Delta\theta_i})\rangle\right), \tag{2.16}$$

where $\langle U\rangle$ denotes the expectation value of U acting on some initial state and under measuring a given observable, and $\boldsymbol{\Delta\theta_i}$ is a vector of the same length as

$\boldsymbol{\theta}$ that is $\frac{\pi}{4r}$ at the i-th position, and zero everywhere else. In other words, the partial derivative of a PQC can be computed by the difference of two evaluations of the same PQC, with the parameter that is considered for the derivative shifted by $\frac{\pi}{4r}$ and $-\frac{\pi}{4r}$, respectively. The factor r depends on the eigenvalues of the generator as $r = \frac{a}{2}(e_1 - e_0)$ [40], and $r = \frac{1}{2}$ for the Pauli gates we commonly use in this thesis [39]. In the case that one parameter is shared between a number of gates, the parameter-shift rule has to be applied for each of the gates individually, and the derivative is then computed according to the product rule. The authors of [33] also show how the above can be extended for generators with arbitrary eigenvalues, at the cost of introducing one ancilla qubit, and for Gaussian gates in continuous-variable quantum computing. The above parameter-shift rule enables computation of exact derivatives of arbitrary quantum circuits, at the cost of two circuit evaluations per parameterized gate in each update step. These gradients can then be used to perform any gradient-based update routine, like the stochastic gradient descent method described above. The downside of the parameter-shift rule is that the number of circuit evaluations required to compute gradients scales linearly with the number of *parametrized gates*, instead of the *number of parameters* itself as in the finite difference method above.

As we have seen above, computing gradients for PQCs is relatively straightforward. However, there are a number of challenges in the optimization of PQC parameters, as we will describe in the next section.

2.2.1.2 Challenges in the optimization of PQCs

In Section 2.2.1.1, we described how the parameters in a quantum circuit can be optimized by using tools from the classical optimization literature. However, it turns out that there are a number of difficulties when optimization is done in a quantum landscape. A first fundamental issue in the optimization of PQCs was described in [41], where the authors introduce the notion of *barren plateaus*, vast saddle points in quantum circuit training landscapes where first and higher order derivatives vanish. These plateaus result from basic features of the geometry of high-dimensional spaces.

The authors of [41] examine the gradients for quantum circuit training in a model known as random PQCs. For this model, parameter updates for gradient-based optimization are calculated based on the expectation value of an observable measured on a state produced by a parametrized circuit, and the circuits are drawn

from a random distribution. The distribution of circuits amounts to choosing a set of gates uniformly at random, where some of the gates have continuous parameters amenable to differentiation. The authors of [41] show that a sufficient, but not necessary, condition for the vanishing of gradients with high probability is that for any arbitrary division of the circuit into two pieces, the two pieces are statistically independent, and that one of them approximately matches the fully random distribution up to the second moment, or in other words forms an approximate 2-design. Formally, a unitary t-design X is defined as,

$$\frac{1}{|X|}\sum_{U\in X} U^{\otimes t}\otimes (U^*)^{\otimes t} = \int_{U(d)} U^{\otimes t}\otimes (U^*)^{\otimes t} dU, \tag{2.17}$$

where the unitary t-design is an ensemble of unitaries that matches the fully random distribution of unitaries up to the t-th moment. The fully random distribution of unitaries is given by the *Haar measure*, which assigns a translation invariant volume on the sphere. To understand the above, it is important to note that in order to sample unitaries of a given size *uniformly* at random from the full Hilbert space, it is not enough to choose an appropriate parametrization, e.g., parametrized unitaries that represent arbitrary rotations, and then sample the *parameters* uniformly at random. This is because those unitaries act on a high-dimensional sphere, where differences between parameters are not proportional to the differences between the resulting points on the sphere, as these differences depend on their position on the sphere. The Haar measure assigns a volume on the sphere that is invariant to the position, and therefore sampling uniformly at random according to the Haar measure allows to sample unitaries uniformly at random from the full Hilbert space. Additionally, a t-design over unitaries generates a t-design of states in the given Hilbert space. While executing a circuit that implements such a Haar-random unitary takes time exponential in the number of qubits, there are efficient techniques to produce circuits that approximate the first and second moment of the Haar distribution, which is formalized by the notion of 2-designs. It has been shown in [42] that random PQCs form approximate 2-designs once they reach a certain depth, and the authors of [41] show how even a modest number of qubits and layers of random gates is enough for this. The depth of a quantum circuit required to reach this regime of barren plateaus depends on the number of qubits and allowed connectivity. It is thought that the depth required to reach a 2-design scales roughly as $\tilde{O}(n^{1/D})$, converging to a logarithmic required depth in the all-to-all connectivity limit [43], where D is the dimension of

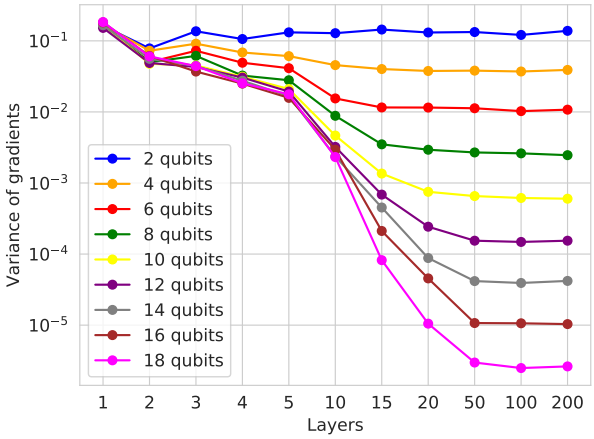

Figure 2.3: Concentration of variance of gradients of the expectation value of the readout qubit. For random parametric quantum circuits, as circuits of different sizes converge to a 2-design, gradient values necessary for training vanish with increasing number of qubits and layers.

the connectivity of the device, e.g., $D = 2$ for a square lattice, and n is the number of qubits.

As an approximation of a 2-design, a particular class of random circuits that were studied numerically in [41] were those with some discrete structure of gates determined by an underlying geometry (e.g., 1D line, 2D array, or all-to-all), and single qubit gates with a continuous parameter rotating around a randomly chosen axis. This assigns a parameter θ_i to each of these gates. To sample from the distribution of random circuits, each angle θ_i was drawn from the uniform distribution $\theta_i \in \mathcal{U}(0, 2\pi)$. Due to concentration of measure effects as described in [41], random PQCs with different sets of parameters will produce very similar outputs and their variance vanishes for sufficiently large circuits, i.e., those that reach approximate 2-designs, as shown in Figure 2.3. Each data point in the figure is calculated over 1000 randomly generated circuits with all-to-all connectivity in each layer, and an initial layer of Hadamard gates on each qubit, to avoid a bias of gradients with respect to the circuit structure caused by initializing in an all-zero state. Note that the average value of the gradient here is zero in all cases.

This represents a generic statement about the volume of quantum space for such

a circuit where one expects to find trivial values for observables. In other words, sufficiently deep, arbitrary random PQCs will produce very similar expectation values regardless of the set of individual parameters. Consequently, the partial derivatives of an objective function based on expectation values of a random PQC will have extremely small mean and variance. These properties make them essentially untrainable for gradient-based or related local optimization methods on quantum devices, as the training landscapes in these scenarios are characterized by large regions where the gradient is almost zero, but which do not correspond to a local or global minimum of the objective function. This decay of the variance of gradients is not only detrimental to parameter optimization, but more generally hinders extraction of meaningful values from quantum hardware, especially on near-term processors that are subject to noise.

Based on the above results by [41], trainability issues in PQCs have been studied extensively in the past few years. While the above results illustrate how with increasing depth, a PQC gets closer to the 2-design regime and is therefore more susceptible to barren plateaus, it was shown in [44, 45] that barren plateaus can also be present in shallow circuits when the cost function is global, in the sense that it is computed based on states in exponentially large Hilbert spaces, instead of local, where the cost function depends only on smaller subsets of the qubits in a circuit. It has also been shown that there is a relationship between the amount of entanglement in a given circuit and its susceptibility to barren plateaus, where high amounts of entanglement tend to lead to untrainability [46, 47]. Moving away from only considering circuits that are approximate 2-designs, the authors of [48] establish a connection between the expressivity of a PQC and barren plateaus, where the expressivity of a circuit is measured in terms of its distance to a 2-design. Finally, it was shown in [49, 50] that regardless of the specific structure of the ansatz, the noise present on quantum hardware can lead to untrainability issues for circuits where the depth grows at least linearly with the system size. While the above results focus on gradients and their variance, it has also been shown that the barren plateau phenomenon persists for higher-order derivatives [51], as well as gradient-free optimization [52]. The amount of negative results above seems discouraging. However, there has also been a tremendous effort to develop methods to address the above issues. The authors of [53] introduce a non-random parameter initialization strategy, where some parts of the circuit are initialized randomly, and the rest is then chosen so that the whole circuit will act as the identity. This will prevent initialization on a barren plateau. Another approach

that addresses the barren plateau issue by choosing a specific parametrization is that by [54], where a number of gates in a circuit share the same parameter and are therefore correlated. Other approaches focus on the circuit structure itself, instead of the parametrization of gates. The authors of [55] introduce an ansatz that is tailored to a specific type of learning problem, and a subsequent work shows that this problem-related structure prohibits the onset of barren plateaus in this ansatz [56]. In a similar vein, the authors of [57] show that circuits that preserve a certain symmetry are immune to barren plateaus. Another recent approach uses techniques related to shadow tomography to detect and avoid barren plateaus [58]. In Chapter 4 of this thesis, we will introduce another method to address the barren plateau problem, where the circuit is initialized and trained in a way that avoids utilizing the full Hilbert space, and thereby sidesteps going into the 2-design regime.

To summarize, there are a number of fundamental challenges in the training of PQCs which present large hurdles that have to be overcome to train VQAs on a large scale. At the same time, addressing and mitigating these challenges is a prosperous field of research, and recent results provide hope that especially problem-tailored ansatzes will enable successful training of large scale quantum models in the future.

2.2.2 Application areas and outlook

The VQA framework is extremely flexible and can be applied to a wide variety of problems. However, three main application areas of VQAs have gained traction in the past years: combinatorial optimization, quantum chemistry and simulation, and machine learning. For combinatorial optimization, the most commonly used technique is the quantum approximate optimization algorithm (QAOA) [59], where the solutions of an optimization problem are encoded as the eigenstates of a Hamiltonian, and the ground state represents the optimal solution. The parameters of the PQC are then optimized such that the circuit outputs the ground state with highest probability. In the limit of an infinitely deep circuit, this algorithm also has theoretical guarantees to find the ground state. Next to being studied in-depth in the context of the canonical Max-Cut problem [60, 61, 16, 62, 63], the QAOA has also been applied to a number of industrially relevant problems [64, 65, 66, 67], as well as being studied experimentally on a superconducting quantum device [68] and with Rydberg atoms [69].

A similar approach to find ground states of Hamiltonians in a quantum chemistry setting is called the variational quantum eigensolver (VQE) [18, 20]. Here, the goal is still to find the ground state of a given Hamiltonian, but the structure of the ansatz to do this is not given in advance as in the case of the QAOA, but can be chosen depending on the given problem. This approach has also been studied extensively in the past few years, both theoretically [70, 71, 72], as well as experimentally on real hardware [73]. The third application area where VQAs have gained much interest in recent years, and the one we focus on in this thesis, is machine learning. Similarly to the above two examples, there has been a wealth of results in this area in the past years, and we will discuss VQAs in this context in more detail in Section 3.3.

Overall, VQAs seem to be a promising road to demonstrate the usefulness of quantum computers on a task of practical interest. However, this road is not without obstacles. In addition to the challenges we described in Section 2.2.1.2, and even when VQAs turn out to be applicable to large-scale quantum systems, the question remains whether these algorithms are better than their classical counterparts. Due to their variational nature, it is hard to make rigorous statements about any type of quantum advantage for VQAs. Second, even if a task with a potential empirical gain in performance is found, it is hard to directly compare these types of algorithms to their classical analogs. In particular, there is the question which classical algorithm one compares to, and under which criteria. To give an example, it has been empirically observed in a number of studies that PQCs seem to be able to solve certain tasks with fewer parameters than classical neural networks [74, 75, 76, 77]. However, one can not directly proclaim quantum advantage from this fact, as it also has to be taken into account that the computation of gradients is more efficient in the classical setting and can be heavily parallelized, which is not the case in a VQA. And last but not least, while VQAs have been conceived with NISQ devices in mind, they too do suffer from the noise present on these devices. An open question is how one can efficiently combat these types of errors. While a number of error mitigation techniques have been proposed in the past years [78, 79, 80], recent results also show that they come with an exponential sampling overhead [81, 82]. So even if VQAs eventually turn out to be useful for practical tasks in the future, a number of roadblocks still have to be overcome until we get there.