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

Skolik, A.

CHAPTER 7

Robustness of quantum reinforcement learning under hardware errors

One of the reasons that VQAs have gained increased interest in the past years is that their hybrid nature, where a large part of the computation is offloaded to a classical device, is hypothesized to make them robust to quantum hardware noise to some extent [261, 14]. This hypothesis is also inspired by classical neural networks, which are robust under certain types of noise. In the classical setting, one can broadly distinguish between two types of noise: benign noise that does not severely impact the training procedure or can even improve generalization [262, 263, 264, 265], and adversarial noise which is deliberately constructed to study where neural networks fail [266, 267, 268, 269]. Furthermore, we can distinguish between noise that is present during training, and noise that is present when using the trained model. Adversarial noise is usually of the latter case, where a trained neural network can produce completely wrong outputs due to small perturbations of the input data [270]. The benign type of noise mentioned above on the other hand is usually present at training time in form of perturbations of the input data, activation functions, weights or structure of the neural network, and has even been established as a method to combat overfitting in the classical literature [262, 263, 264, 265, 271].

These results inspired the hypothesis that variational quantum algorithms possess a similar robustness to certain types of noise and may even benefit from its presence when trained on a quantum device. However, thorough investigations that confirm such robustness of VQAs against hardware-related noise, or even a beneficial effect from it, are still lacking. In terms of negative results for the trainability of VQAs under noise, it has been shown that optimization landscapes of noisy

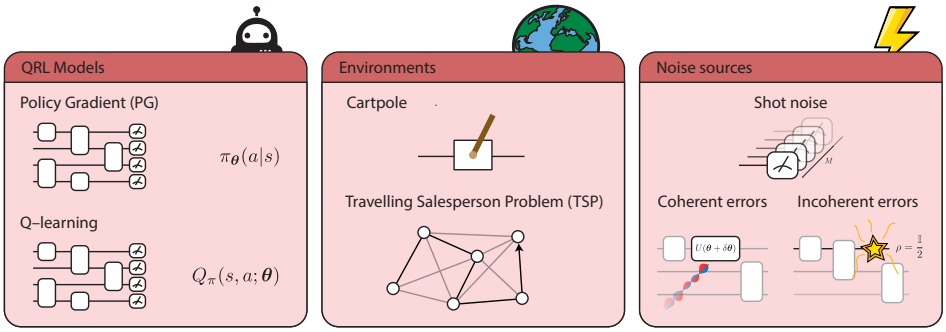


Figure 7.1: Summary of the scenarios analysed in the present work. We consider two models for quantum reinforcement learning (QRL) agents and test their performances on two environments, CartPole and the Travelling Salesperson Problem (TSP). We analyse the performances of the agents when these are trained and used in the presence of most common noise sources found on real quantum hardware, namely statistical fluctuations due to shot noise, coherent errors due to imperfect control or calibration of the device, and finally incoherent errors coming from the unavoidable interaction of the quantum hardware with its environment.

quantum circuits become increasingly flat at a rate that scales exponentially with the number of qubits under local Pauli noise when the circuit depth grows linearly with the number of qubits [49]. In the case of the variational quantum eigensolver, where the goal is to find the ground state of a given Hamiltonian, the presence of noise has been shown to lead to increasing deviation from the ideal energy [272]. Similar effects have been studied in the context of the quantum approximate optimization algorithm (QAOA) [59], where the goal is to find the ground state of a Hamiltonian that represents the solution to a combinatorial optimization problem [273, 68].

When it comes to QML, in-depth studies on the effect of noise on the trainability and performance of VQAs are scarce. Apart from the work mentioned above on noise-induced barren plateaus [49], the authors of [166] provided first insights into how the data encoding method used in a quantum classifier influences its resilience to varying types of noise. As for the potential benefit of noise, the authors of [274] show that the stochasticity induced by measurements in a QML model can help the optimizer to escape saddle points. The above results show that, on the one hand, too much noise will make the model untrainable, while on the other hand, modest amounts of noise can even improve trainability [274]. However, it remains

unclear how large the gap is between tolerable and harmful amounts of noise [261], and it is not expected that this can be answered in a general way for all different types of learning algorithms and noise sources.

In this chapter, we shed light on this question from the angle of variational quantum reinforcement learning. Classical reinforcement learning models have been shown to be sensitive to noise, either during training [275] or in the form of adversarial samples [276, 277]. Additionally, it is known that a bottleneck of RL algorithms is their sample inefficiency, i.e., many interactions with an environment are needed for training [278]. Still, RL resembles human-type learning most closely among the main branches of modern ML, and therefore motivates further studies in this area. Among these studies, RL with VQAs has been proposed and extensively investigated in the noise-free setting over the past few years [153, 154, 150, 75, 157, 199, 76, 279, 156]. These results provide promising perspectives, as quantum models have empirically been shown to perform similarly to neural networks on small classical benchmark tasks [75, 76], while at the same time an exponential separation between classical and quantum learners can be proven for specific contrived environments based on classically hard tasks [150, 75]. These results motivate further studies on how large the above-mentioned gap between tolerable and too much noise is in the case of variational RL algorithms, and how close the algorithm performance can get to the noise-free setting for various types of noise that can be present on near-term devices.

We investigate this for two types of variational RL algorithms, Q-learning and the policy gradient method, by performing extensive numerical experiments for both types of algorithms with two different environments, CartPole and the Travelling Salesperson Problem, and under the effect of a wide class of noise sources, namely shot noise, coherent and incoherent errors. In Figure 7.1 we summarise the approach of the present work showing the QRL models, environments and noise sources considered in the analysis. We start by considering the trade-off between the number of measurement shots taken for each circuit evaluation and the performance of variational agents. As the number of shots required by a QML algorithm can be a bottleneck on near-term devices and RL is known to require many interactions with the environment to learn, we propose a method for Q-learning to reduce the number of overall measurements by taking advantage of the structure of the underlying RL algorithm. Second, we model coherent errors with a random Gaussian perturbation of the variational parameters, and analytically study the

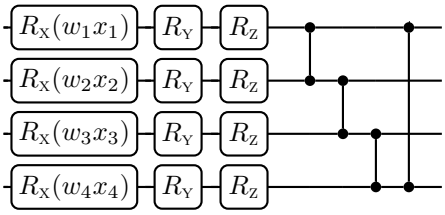
effect of these perturbations on the output of parameterised quantum circuits, similarly to [280]. We provide an upper bound on the perturbation induced by such Gaussian coherent noise based on the Hessian matrix of the circuit, and theoretically and numerically show that hardware-efficient ansätze may be particularly resilient against this type of error due to small second derivatives [51]. Finally, we analyse the performance of the above algorithms under the action of incoherent errors coming from the unavoidable interaction of the qubits with the environment which we have no control over. To study this type of noise, we start by investigating the effect of single-qubit depolarization channels. In addition, we consider a custom noise model that combines various types of errors present on hardware, and study the effect of this noise model with error probabilities that are present in currently available superconducting quantum hardware. Our results show that both policy gradient methods and Q-learning exhibit a robustness to noise that may enable successfully running them on near-term devices. This motivates further study in the quest to find a real-world problem of interest where a quantum advantage for variational RL could be possible.

7.1 Environments and implementation

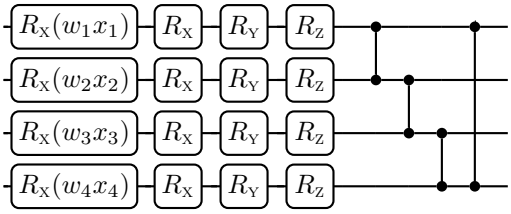
Our goal is to get insight into the effect of noisy training on quantum RL algorithms. For this, we consider quantum versions of the two main paradigms in RL that have been introduced in previous sections: value-based methods (see Section 3.2.2) and policy gradient methods (see Section 3.2.3). As we are interested in the effect of noisy training on models that have otherwise been proven to work well in the noise-free setting, we study models and environments that have been already investigated in this setting before [150, 75, 174]. In this way, we have evidence that the models and hyperparameters that we choose are suitable for the studied environments, and can focus our efforts on understanding the effect that noise has on the training and performance of these agents. The code that was used to generate the numerical results in this work can be found on Github [281].

7.1.1 CartPole

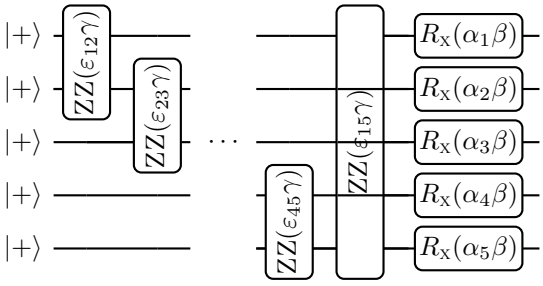
The first environment that we study is the CartPole environment from the OpenAI Gym [282] that was also studied in Chapter 5. For a detailed description of the environment and the implementation of the quantum Q-learning agent, we refer the reader to Section 5.1. For the policy gradient method, we follow the



(a) HWE-Q



(b) HWE-PG



(c) EQC

Figure 7.2: Parameterised circuits used in this work. (a) Hardware-efficient ansatz for Q-learning in the CartPole environment from [75], (b) hardware-efficient ansatz for policy gradient method in the CartPole environment from [150], (c) equivariant quantum circuit for Q-learning and policy gradient method in the TSP environment from [174]. For (a) and (b) we use 5 repetitions of the template shown above, while for (c) we use just one layer.

implementation used in [150] and made available at [283], which uses five layers of the same hardware-efficient ansatz used for the Q-learning agent, except that each layer has an additional trainable rotation around the x-axis on each qubit (see Figure 7.2(b)), and the action observables are defined as $O_L = Z_1Z_2Z_3Z_4$ and $O_R = \mathbb{I} - O_L$. As before, input features are multiplied with an additional trainable parameter each. Since the policy is a probability distribution, a final SoftMax layer is used to map the expectation values $\langle O_a \rangle_{s,\boldsymbol{\theta}} \in [-1, 1]$ to the appropriate range $[0, 1]$, and so probabilities for each action eventually become

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{e^{\beta \langle O_a \rangle_{s,\boldsymbol{\theta}}}}{\sum_{a'} e^{\beta \langle O_{a'} \rangle_{s,\boldsymbol{\theta}}}}, \tag{7.1}$$

where $\beta \in \mathbb{R}$ is a also a trainable parameter. A depiction of both circuits can be seen in Figure 7.2 a) and b).

7.1.2 Traveling Salesperson Problem

The second environment that we study is the TSP environment from Chapter 6, where again the Q-learning agent and the environment are implemented as described in Section 6.2 and the ansatz can be seen in Figure 7.2 c). For policy gradient agents the ansatz is the same as in the Q-learning case, but as the policy has to be a probability distribution we again use a final SoftMax layer with a trainable inverse temperature β on the observable, as in Equation (7.1). The authors of [150] have shown that using this type of final layer can be highly beneficial for policy gradient training, compared to only using the probability distribution resulting from the quantum state directly. This is due to the fact that the trainable inverse temperature enables the agent to tune its level of exploration of the state space. As the optimal solutions to TSP instances are deterministic, it is favourable in this environment to have a tunable inverse temperature that allows exploration of the large state space early in training, as well as close-to-deterministic decisions towards the end.

7.2 Shot noise

We start our studies with the type of noise that is arguably the simplest to characterize: noise induced by statistical errors that result from the probabilistic nature of quantum measurements. For each circuit evaluation, be it for action selection of the RL agent or for computing parameter updates via the parameter

shift rule, we take a fixed number of measurements M and compute the resulting expectation value. The precision ϵ of this expectation value depends on M and scales as $\epsilon \sim 1/\sqrt{M}$, as we will explain in more detail below.

Variational algorithms often require a very large number of measurements to be executed, and this problem is exacerbated in QML tasks that typically involve separate circuit evaluations for all training data points. For this reason, it is not only important to understand the effect of shot noise on the trainability and performance of QML models, but it is also desirable to develop methods that lead to a smaller shot footprint than simply assigning a fixed number of shots to each circuit evaluation. Depending on knowledge of the algorithm itself, it can be possible to make an informed decision on the number of shots that suffice in each step. In this section, we develop such a method specifically for Q-learning that is a natural extension to the original algorithm.

7.2.1 Reducing the number of shots in a Q-learning algorithm

As described in Section 3.2.2, a Q-learning agent selects actions based on the following rule (see Equation (3.18))

$$a_t = \text{argmax}_a Q_\pi(s_t, a; \boldsymbol{\theta}),$$

that is, it chooses actions according to the largest Q-value.[1] Now, consider a quantum agent that only has access to noisy estimates of the Q-values $\tilde{Q}(s_t, a_t; \boldsymbol{\theta})$ resulting from the statistical uncertainty of a measurement process involving a finite number of shots M. If the sample size is large enough $M \gg 1$, then by the central limit theorem each noisy Q-value can be described as a random variable

$$\tilde{Q}(s_t, a_t; \boldsymbol{\theta}) = Q(s_t, a_t; \boldsymbol{\theta}) + \epsilon\,, \tag{7.2}$$

where $Q(s_t, a_t; \boldsymbol{\theta})$ is the true noise-free value, and ϵ is a random variable sampled from a Gaussian distribution centered in zero $\mu_\epsilon = \mathbb{E}[\epsilon] = 0$, and with standard deviation inversely proportional to the square root of the number of measurement shots $\sigma_\epsilon = \text{Std}[\epsilon] \sim 1/\sqrt{M}$. Since actions are selected through an argmax function,

[1] In the ϵ-greedy policy (see Section 3.2.2) we consider here, the agent picks either the action corresponding to the argmax Q-value, or a random action. As no circuit evaluation is required to pick a random action, we only consider the steps with actual action selection by the agent in this section.

the perturbation ϵ will not affect the action selection process as long as the order between the largest and the remaining Q-values remains unchanged. Then, one may ask: is there a minimal number of shots that suffice to reliably distinguish the largest Q-value Q_{max} and the second-largest Q-value Q_2?

When the observables associated to the actions are non-commuting, they have to be estimated independently from each other, and one has the freedom of choosing how to allocate the measurement shots among the observables of interest, possibly in a clever way. In our case, the goal is to estimate which of the observables has the highest Q-value while trying to be shot-frugal, and this task can be related to the theory of multi-armed bandits [284]. The multi-armed bandit is a RL problem in which an agent can allocate only a limited amount of resources between a number of choices, e.g., a number of arms on a bandit machine, and is asked to determine which of these choices leads to the highest expected reward. There exists a trade-off between exploration (i.e., trying the different arms) and exploitation (always choosing the arm that appears best according to the current knowledge), and the upper confidence bound (UCB) [285, 286] algorithm shows how to use statistical confidence bounds to allocate exploratory resources. The UCB algorithm could be used in the scenario described above where a number of non-commuting observables have to be estimated, and we want to find the optimal strategy to allocate a fixed budget of measurement shots to the task of identifying the largest Q-value.

However, in the specific implementations of QRL agents based on recent literature that we study in this work [150, 75, 174], only commuting observables are used, hence it is not necessary to apply the UCB procedure to determine which one should be measured more often. Nonetheless, inspired by the UCB algorithm, we can still define a rather general simple heuristic that can be used to reduce the overall number of shots required to train the Q-learning models as those studied in this work. The idea is to use the knowledge about the scaling of the estimation error with respect to the number of measurements (see Equation (7.2)), to determine with confidence whether we have taken enough shots to determine the maximum Q-value.

The procedure goes as follows. First, we take a small number of initial measurements m_{init}, for example $m_{\text{init}} = 100$, of all observables to compute the estimates $\tilde{Q}_{m_{\text{init}}}(s_t, a),\ \forall\ a \in \mathcal{A}$. Based on these values, we compute the absolute difference between the largest and the second largest Q-values. If this difference is larger

Algorithm 1 Algorithm to reduce the number of measurements in Q-learning

Input m_{init}, m_{inc}, m_{max}

Output m_{est}

```
m_est ← m_init
while m_est < m_max do
    Q̃(s_t, a_1) = ⟨O_{a_1}⟩_{m_est}
    Q̃(s_t, a_2) = ⟨O_{a_2}⟩_{m_est}
    ΔQ̃ = |Q̃(s_t, a_1) − Q̃(s_t, a_2)|
    if ΔQ̃ < 2/√m_est then
        m_est ← m_est + m_inc
    else
        return min(m_est, m_max)
    end if
end while
return min(m_est, m_max)
```

than twice the estimation error $\epsilon = 2/\sqrt{m_{\text{init}}}$ (as both of the Q-values are noisy), we have found the largest Q-value with high confidence and we stop here. On the other hand, if the difference is smaller, we increment the sample size with additional m_{inc} measurements each, and recompute the estimated Q-values with the $m_{\text{inc}} + m_{\text{init}}$ shots. We again compute the absolute difference of the two largest Q-values and determine whether the number of measurements suffices based on the error $\epsilon = 2/\sqrt{m_{\text{init}} + m_{\text{inc}}}$. This measure-and-compare scheme is performed until either the two largest Q-values can be distinguished with high confidence, or a fixed shot budget m_{max} is reached. In Algorithm 1 we provide a description of this procedure, where for the sake of simplicity we describe the case where there are only two possible actions, and we therefore only have to find the larger of two Q-values. However, the scheme can be used for an arbitrary number of Q-values, as it is only important to distinguish between the highest and the second-highest Q-value with high confidence. The algorithm takes as input the number of initial measurements m_{init}, the number of additional measurements in every step m_{inc}, and the maximum number of measurements that are allowed in one run of the shot-allocation algorithm (i.e., finding the largest Q-value) m_{max}. The output is the number of measurements m_{est} that are sufficient to find the argmax Q-value with high confidence based on the rules above. The values $\langle O_{a_i} \rangle_{m_{\text{est}}}$ are the expectation values of observables O_{a_i} corresponding to action a_i, estimated

with m_{est} shots. Note that the proposed scheme works both for commuting or non-commuting observables, where in the former case one can spare shots by computing the observables from the same set of measurement outcomes. Moreover, note that we ignore the coefficients in the statistics of the Q-values coming from **??**, when considering the measurement stopping criterion. This choice has no impact on the effectiveness of the proposed method, as it is always found to be very well performing in the presented form.

While this algorithm can clearly determine the optimal number of shots in the action selection process in a methodical manner, one should check that this will not introduce errors in the remaining parts of the variational Q-learning model, i.e., during the parameter update step. Recall that each parameter update of the model is computed based on the output of the model itself (see Equation (3.19))

$$Q_\pi(s_t, a_t; \boldsymbol{\theta}) \leftarrow r_{t+1} + \gamma \max_a Q_\pi(s_{t+1}, a; \boldsymbol{\theta}),$$

which means that in the parameter update step we do not need to perform action selection, but instead care about the actual Q-values in order to compute the loss function. The question is now to what precision we need to approximate the Q-values in order to learn a good Q-function. Technically, even the noise-free Q-function is only an approximation of the true Q-function, which is the whole point of doing Q-learning with function approximators. This suggests that there is some leeway to make even the approximate function itself an approximation by taking only as many measurements as are necessary to find the argmax Q-value with high confidence. Indeed, it has been shown in [75] that even the Q-functions of agents that successfully solve an environment can produce Q-values that are far from the optimal Q-values, and that learning the correct order of Q-values is more important in this setting than approximating the optimal Q-value as precisely as possible. Consequently, when we compute the Q-values that are used to perform parameter updates, we use the same algorithm as that in Algorithm 1 to determine the number of measurements to take.

7.2.2 Numerical results

We now numerically compare the performance of agents in the CartPole and TSP environments in settings where a fixed number of shots is used in each circuit evaluation, and where the number of shots in each step is determined by the algorithm we introduced in Section 7.2.1. To give an overview of the number of

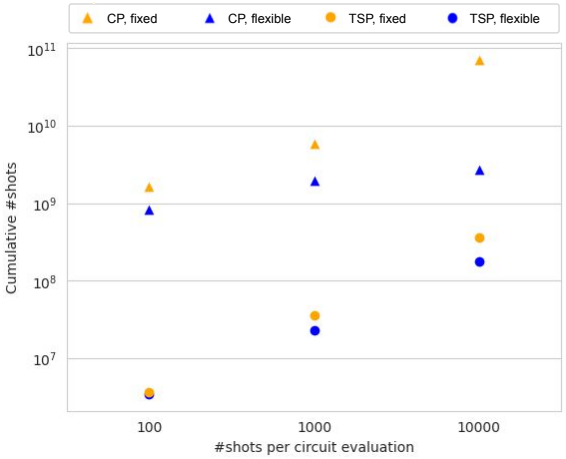


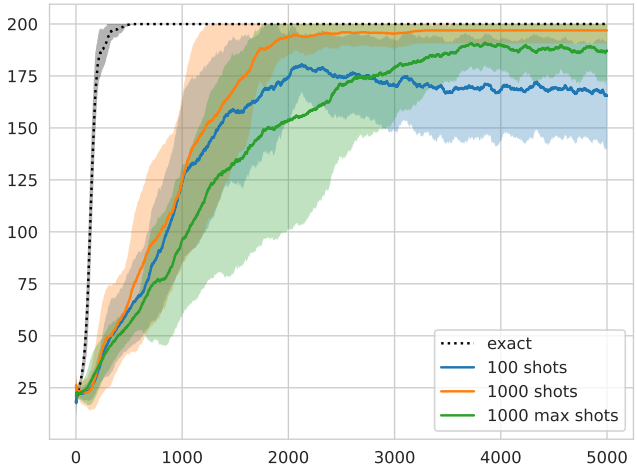
Figure 7.3: Comparison of the cumulative number of shots per observable over a full training run, for the flexible shot allocation technique (blue) and for a standard fixed measurement scheme using the same number of shots for every circuit evaluation (orange), both for CartPole (triangles) and TSP (circles). Each data point shows the average over ten trained agents.

shots used in one training run under varying hyperparameter settings, we show the average cumulative number of shots for different settings in Figure 7.3. For the CartPole environment (triangles), the number of cumulative shots grows quickly with the number of shots in each step in the fixed setting (orange). This is not true for the flexible shot allocation technique (blue), where for values of $m_{\max} \in \{100, 1000, 10000\}$ the cumulative number of shots is relatively similar. As we see in Figure 7.4 a), a low number of shots such as 1000 is already sufficient to achieve close to optimal performance in the CartPole environment. Therefore, we focus on comparing settings with 100 and 1000 (maximum) shots per circuit evaluation in that figure. Comparing the cumulative number of shots for $m_{\text{fixed}} = 100$ and $m_{\max} = 1000$ in Figure 7.3, we see that these two configurations use almost the same number of measurements overall. Still, the final performance of the agents trained with the flexible shot allocation technique is almost optimal, while those trained with a fixed number of shots in each circuit evaluation are below a final score of 175 on average. However, as we allow agents to use even less than 100 shots per evaluation with the flexible allocation method of Algorithm 1, performance starts to degrade, so at least 100 shots are required in this setting. To not clutter

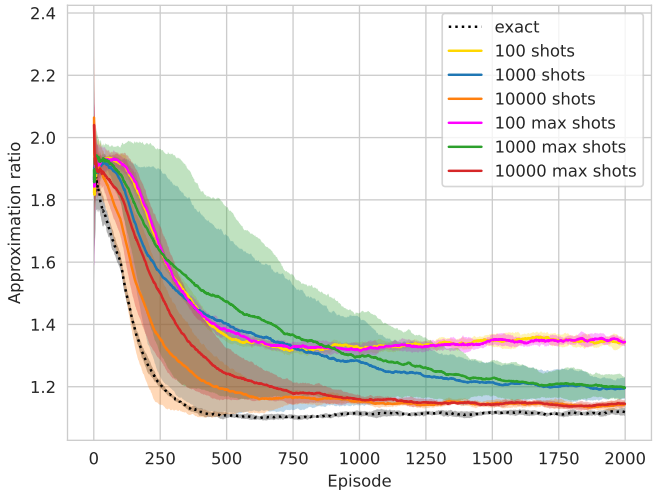
the figure we show the results for agents that use fewer than 100 shots per circuit evaluation in Figure 6 in the Appendix.

In the TSP environment, each step in an episode constitutes of a constant and (compared to CartPole) relatively low number of circuit evaluations. We still see that the higher the setting for the (maximum) number of shots is, the bigger the gap in average cumulative number of shots becomes. For agents trained in the TSP environment, shown in Figure 7.4 b), the final performance remains unchanged by the additional noise introduced by the flexible shot allocation technique, and agents reach the same accuracy of those trained with a corresponding but fixed number of shots per circuit evaluation. The only difference between the two approaches is that the agents using the flexible shot allocation method take slightly longer to converge in some cases. Independently from the estimation method used (flexible or fixed), it is clear from Figure 7.4 that it is the number of shots available that plays the major role in determining the performance of the noisy agents, as measured by the proximity to the average approximation ratios reached in the noise-free scenario, namely when agents have access to exact the expectation values ($M \to \infty$). In this environment, there is a trade-off between delayed convergence due to less precision in the approximation of the Q-function, and using a higher number of shots to arrive at the same final performance.

To summarize, we have seen that Q-learning models can be successfully trained even in the presence of statistical noise introduced by a measurement processes carried out with a limited number of shots. In addition, by leveraging the specifics of the Q-learning algorithm, we introduced an easy-to-implement and effective method that can be used to reduce the number of shots needed to train variational Q-learning agents. How many shots one can save during training with this method depends on the agents' resilience to shot noise, as well as the specific characteristics of the environment. In the CartPole environment, where one bad decision does not lead to immediate failure, the additional noise introduced by estimating expectation values with a low number of measurements and approximating an imprecise Q-function does not affect performance severely. In the TSP environment on the other hand, where one bad choice of the next city in the tour can lead to a much longer path, we observe that the number of measurements has to be relatively high to get close to optimal performance. However, even in this setting we can achieve a reduction in the overall number of measurements by taking an informed approach at when to measure an observable more often.



(a) CartPole



(b) Traveling Salesperson Problem

Figure 7.4: Comparison of Q-learning with shot noise using the informed shot-allocation method (labeled “max shots") proposed in this work, and a standard measurement scheme that simply assigns a fixed number of shots to each circuit evaluation (labeled “shots"). Results are averaged over ten agents for each configuration. (a) Shows results for agents trained in CartPole environment, (b) shows results for agents in the TSP environment.

7.3 Coherent noise

In this section, we turn our attention to coherent noise, that is, errors that preserve the unitary evolution of the quantum circuit but still change its output [287]. In our analysis, we model coherent noise as an *over-* or *under-rotation* of the parametrized gates, by adding a random Gaussian perturbation to the variational parameters in the considered circuits.

This type of error could occur in real quantum devices as a drift in the parameters for example due to an imperfect control of the system or a miscalibration of the hardware, and it is therefore an important component of the overall picture of an imperfect quantum device. Specifically, we assume that the perturbation remains unchanged during the estimation a given observable, i.e. it does not change considerably between repeated measurements on the same experimental setup. However the perturbation amount change whenever the experiment is changed, for example due to measuring a different observable, or using the circuit with a different set of parameters.

Gaussian coherent noise is also an interesting model because it lends itself very well to theoretical analysis, and one can estimate the effect of such an error on the output of a parameterised quantum circuit. In the following, we first proceed with an analytical treatment of the error introduced by Gaussian perturbations on variational circuits, and then proceed with the numerical results for the two environments considered in this work.

7.3.1 Effect of Gaussian coherent noise on circuit output

Consider a general parametrized quantum circuit acting on a system of n qubits, with unitary $U(\boldsymbol{\theta}) \in \mathbb{C}^{2n} \times \mathbb{C}^{2n}$ and parameter vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_M) \in \mathbb{R}^M$. Let O be on observable and $\rho = |0\rangle\langle 0|$ the initial state of the quantum system, the outcome of the variational circuit is the expectation value

$$f(\boldsymbol{\theta}) = \langle O \rangle_{\boldsymbol{\theta}} = \mathrm{Tr}\left[O\, U(\boldsymbol{\theta}) \rho U^\dagger(\boldsymbol{\theta})\right]. \tag{7.3}$$

Suppose that the parameters are affected by a noise process that adds a perturbation

$$\boldsymbol{\theta} \to \boldsymbol{\theta} + \delta\boldsymbol{\theta}\,, \tag{7.4}$$

where $\delta\boldsymbol{\theta} = (\delta\theta_1, \ldots, \delta\theta_M) \in \mathbb{R}^M$ are *i.i.d.* according to a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ with zero mean $\mu = 0$ and equal variance σ^2, namely

$$\begin{aligned} &\delta\theta_i \sim \mathcal{N}(0, \sigma^2)\,, \\ &\mathbb{E}[\delta\theta_i] = 0\,, \qquad\qquad \forall i \in \{1, \ldots, M\} \\ &\mathbb{E}[\delta\theta_i \delta\theta_j] = \sigma^2 \delta_{ij}\,. \end{aligned} \tag{7.5}$$

As discussed earlier, in our analysis in this section and in the numerical simulations in section 7.3.3.1, we assume that the perturbed parameters remain the same during the evaluation of a single expectation value. In a real experiment on quantum hardware, this would mean that for all measurements used to estimate the expectation value, the perturbations stay at least approximately unchanged. Of course, without this assumption, the resulting noise model could not be considered unitary, and one may then resort to a noise channel formulation of Gaussian noise as proposed in [261, 280]. Hence, in the following we restrict our attention to the setting described above.

The effect of Gaussian noise on the circuit can be evaluated by Taylor expanding the circuit around the unperturbed parameters $\boldsymbol{\theta}$. For ease of explanation, we hereby report only the main ideas and results, and we refer to Appendix 8 for a complete and detailed derivation of all the calculations performed in this section.

Let $f(\boldsymbol{\theta} + \delta\boldsymbol{\theta})$ be the function evaluated on the perturbed parameters, its Taylor expansion up to fourth-order reads

$$\begin{aligned} f(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) \approx & f(\boldsymbol{\theta}) + \sum_{i=1}^{M} \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_i} \delta\theta_i + \frac{1}{2} \sum_{i,j=1}^{M} \frac{\partial^2 f(\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} \delta\theta_i \delta\theta_j \\ & + \frac{1}{3!} \sum_{i,j,k=1}^{M} \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j \partial \theta_k} \delta\theta_i \delta\theta_j \delta\theta_k + \mathcal{O}\big(\delta\theta^4\big)\,. \end{aligned} \tag{7.6}$$

With this expression one can evaluate the expected value of the noisy function $\mathbb{E}[f(\boldsymbol{\theta} + \delta\boldsymbol{\theta})]$ over the distribution of the Gaussian perturbations, $\mathbb{E}(\cdot) = \mathbb{E}_{\delta\theta_i \sim \mathcal{N}(0,\sigma^2)}(\cdot)$. Since every odd moment of a Gaussian distribution vanishes, using

relations (7.5) in the expansion (7.6) one obtains

$$\begin{aligned}\mathbb{E}[f(\boldsymbol{\theta}+\delta\boldsymbol{\theta})] &\approx f(\boldsymbol{\theta}) + \frac{1}{2}\sum_{ij}\frac{\partial f(\boldsymbol{\theta})}{\partial\theta_i\partial\theta_j}\mathbb{E}[\delta\theta_i\delta\theta_j] \\ &\approx f(\boldsymbol{\theta}) + \frac{1}{2}\sigma^2\sum_{ij}\frac{\partial f(\boldsymbol{\theta})}{\partial\theta_i\partial\theta_j}\delta_{ij} \\ &\approx f(\boldsymbol{\theta}) + \frac{1}{2}\sigma^2\,\mathrm{Tr}[H(\boldsymbol{\theta})] + \mathcal{O}(\sigma^4),\end{aligned} \tag{7.7}$$

where $\mathrm{Tr}[H(\boldsymbol{\theta})]$ denotes the trace of the Hessian matrix

$$H_{ij}(\boldsymbol{\theta}) = \frac{\partial^2 f(\boldsymbol{\theta})}{\partial\theta_i\partial\theta_j} \quad i,\, j = 1\ldots, M. \tag{7.8}$$

Thus, the first non-vanishing correction term caused by the noise is proportional to the noise variance σ^2, and the Hessian of the parametrized quantum circuit, which conveys geometric information about the curvature of the function landscape around the unperturbed point $\boldsymbol{\theta}$.

Higher-order terms in the expansion can be evaluated in a similar way, specifically making use of so-called *Wick's relations* for multivariate normal distributions as shown in Appendix 8. If all the derivatives of the function $f(\boldsymbol{\theta})$ are bounded, as it is the case for parametrized quantum circuits, then it is possible to derive an upper bound on the error induced by the perturbations which only depends on the noise strength σ^2 and the total number of parameters M, as we show in the following.

Using the parameter shift rule [33, 39], one can show that any derivative of a parametrized quantum circuit can be expressed as a linear combination of circuit outcomes evaluated at specific points in parameter space [280, 51]. Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M) \in \mathbb{N}^M$ be a multi index keeping track of the order of partial derivatives, define the derivative operator

$$\partial^{\boldsymbol{\alpha}} := \frac{\partial^{|\boldsymbol{\alpha}|}}{\partial\theta_1^{\alpha_1}\cdots\partial\theta_M^{\alpha_M}}\,, \tag{7.9}$$

where $|\boldsymbol{\alpha}| := \sum_{i=1}^{M}\alpha_i$. By nested applications of the parameter shift rule, one can show that

$$\partial^{\boldsymbol{\alpha}} f(\boldsymbol{\theta}) = \frac{1}{2^{|\boldsymbol{\alpha}|}}\sum_{i=1}^{2^{|\boldsymbol{\alpha}|}} s_m\, f(\boldsymbol{\theta}_m)\,, \tag{7.10}$$

where $s_m \in \{\pm 1\}$ are signs, and $\boldsymbol{\theta}_m$ are parameters obtained shifting the parameter vector $\boldsymbol{\theta}$ along different directions. Now, since the measurement outcome of every circuit is bounded by the maximum absolute eigenvalue of the observable, i.e. $|f(\boldsymbol{\theta})| \leq \|O\|_\infty$, consequently it also holds that $|\partial^{\boldsymbol{\alpha}} f(\boldsymbol{\theta})| \leq \|O\|_\infty$ (see Appendix 8). Note that we only consider bounded observables here, like the Pauli operators commonly used in variational RL algorithms [153, 154, 150, 75].

Since all the derivatives of the function are bounded, it is possible to bound every term in the Taylor series and then compute an upper bound to the error caused by the perturbation. In fact, defining the absolute (average) error caused by the noise as

$$\varepsilon_{\boldsymbol{\theta}} := \left|\mathbb{E}[f(\boldsymbol{\theta} + \delta\boldsymbol{\theta})] - f(\boldsymbol{\theta})\right|, \tag{7.11}$$

one can prove that this is upper bounded by (see Appendix 8)

$$\varepsilon_{\boldsymbol{\theta}} \leq \|O\|_\infty \left(e^{\sigma^2 M/2} - 1\right). \tag{7.12}$$

Note that since $\varepsilon_{\boldsymbol{\theta}} \leq 2\|O\|_\infty$ is always true, the bound is informative only as long as $e^{\sigma^2 M/2} - 1 < 2$.

This expression only depends on the noise strength σ^2, the total number of noisy parameters M, and the operator norm of the observable $\|O\|_\infty$, and it can be used to estimate a *sufficient* condition on the noise strength to guarantee a desired error threshold $\varepsilon_{\boldsymbol{\theta}}$. Rearranging Equation (7.12), a sufficient condition to have error $\varepsilon_{\boldsymbol{\theta}}$ not larger than ϵ, is to have Gaussian perturbations satisfying

$$\sigma \leq \sqrt{\frac{2}{M}} \log\left(1 + \frac{\epsilon}{\|O\|_\infty}\right). \tag{7.13}$$

As the allowable error is small $\epsilon \ll 1$, by approximating the logarithm $\log(1+x) \approx x$, one derives that the perturbations must follow the scaling

$$\sigma \in \mathcal{O}\left(\frac{\epsilon}{M^{1/2}\|O\|_\infty}\right). \tag{7.14}$$

Note that a similar scaling law was recently derived also in [280], though via a slightly different method based on the moment generating function of the probability distribution characterising the perturbations.

To provide an example, assume one is willing to tolerate an error of $\epsilon = 10\%$, that $\|O\|_\infty = 1$ as for measuring a Pauli operator and that the PQC consists of $M = 100$ noisy parametrized gates, then one can be sure of such accuracy if

$\sigma \sim 0.1/\sqrt{100} = 0.01$. However, we stress again that the scaling Equation (7.13) is only a *sufficient* but not necessary condition for achieving an error ϵ. In fact, apart from the requirement of bounded derivatives, Equation (7.13) is agnostic with respect to the specifics of the function, and such bound can be quite loose in real instances where a much larger noise level still causes a small error, as shown in Figure 7.5.

In Figure 7.5, we report simulation results obtained by simulating the parametrized ansatz depicted in Figure 7.2(b) subject to Gaussian coherent noise of increasing strength. It is clear that the output of the circuit closely follows the approximation of Equation (7.7) given by the Hessian even at moderately large value of the noise $\sigma \lessapprox 0.15$. When the noise is too strong ($\sigma > 0.2$), the circuit becomes essentially random, and the average expectation value when measuring a Pauli operator is zero. This is a consequence of PQCs often behaving like unitary designs upon random initialization of the parameters [229, 44], a fact which we discuss in detail in Sec. 7.3.2. At last, as discussed earlier, while the upper bound (7.12) holds, it is indeed very loose and only holds tightly at small $\sigma \lessapprox 0.01$.

We now proceed discussing why hardware-efficient parametrized quantum circuits can be resilient to Gaussian coherent noise. Roughly, this is because such circuits are found to behave like random unitaries upon random assignment of the parameters, which implies that the derivatives of such circuits tend to vanish as the system size grows large [51].

7.3.2 Resilience of Hardware-Efficient ansatzes to Gaussian coherent noise

The previous analysis showed that Gaussian perturbations induce an error depending on the Hessian of the circuit (see Equation (7.7)), so that up to fourth order in the perturbation it holds that

$$\mathbb{E}[f(\boldsymbol{\theta} + \delta\boldsymbol{\theta})] \approx f(\boldsymbol{\theta}) + \frac{1}{2}\sigma^2 \operatorname{Tr}[H(\boldsymbol{\theta})]\,. \tag{7.15}$$

This equation tells us that if the optimization landscape is flat or close to being flat, then the Hessian is small, and so the perturbation will have little effect on the output of the circuit. On the contrary, in the presence of a very curved landscape, noise will have a great impact and the output of the circuit may change sensibly. It is known that the curvature of the optimization landscape produced by a PQC is

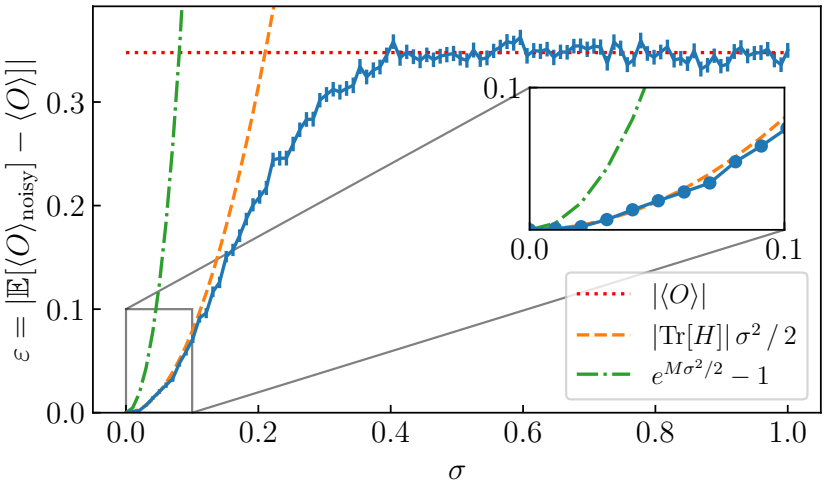


Figure 7.5: Effect of Gaussian coherent noise on the output of the parametrized quantum circuit shown in Figure 7.2(b). The plot is obtained by first choosing a parameter vector $\boldsymbol{\theta}_0 \in \mathbb{R}^{92}$ corresponding to a the ideal noise-free expectation value $f(\boldsymbol{\theta}_0) = \langle O \rangle$ with $O = Z^{\otimes 4}$. With this baseline fixed, random Gaussian perturbations are added to the angles $\boldsymbol{\theta}_{noisy} = \boldsymbol{\theta}_0 + \delta\boldsymbol{\theta}$, and the resulting noisy expectation vales $\langle O \rangle_{noisy}$ are computed. Each point in the plot is the average over $N = 10^5$ different perturbation vectors sampled from a multivariate Gaussian distribution of a given σ. The experiments are then repeated for increasing values of the noise strength σ. The error bars show the statistical error of the mean. For small noise levels, the output of the quantum circuit closely follows the behaviour predicted by Equation (7.7), where the Hessian is evaluated at the unperturbed value $H = H(\boldsymbol{\theta}_0)$. When the error is too large the circuit behaves as a random circuit whose output is on average zero, hence the error plateaus to the unperturbed expectation value $\varepsilon = |\langle O \rangle| = |f(\boldsymbol{\theta}_0)|$). The upper bound predicted by Equation (7.12) is very loose in general, and holds tightly only for very small values of $\sigma \lessapprox 0.01$.

closely related to the barren plateau phenomenon [229, 44, 48], where the variance of the first and second derivative vanishes exponentially in the number of qubits and layers in a random circuit. Additionally, the hardware-efficient ansatz we use for some of the environments in this work is known to suffer from barren plateaus when the system size is large. As the curvature of the optimization landscape of these types of circuits is very flat, it can also be expected that the type of noise induced by the Gaussian perturbations on parameters that we study in this work should not affect circuits that generally produce small first and second order derivatives. While circuits that are in the barren plateau regime are obviously undesirable as they quickly become untrainable, one can consider circuits of the size such that the variance in gradients is relatively small, but the circuit has not yet converged to an approximate 2-design, as shown in [229]. We make this statement more formal in the following.

We can use standard results on averages of unitary designs [288, 289] to characterize the Hessian of hardware-efficient circuits, and thus gain insight on their performance under Gaussian noise. We report the main results of our analysis here, full derivations can be found in Chapter 8. In the following, we suppose that sampling a random value of the parameter vector $\boldsymbol{\theta}$ in the parametrized circuit $U(\boldsymbol{\theta})$, is equivalent to sampling a unitary from a unitary 2-design, defined as a set of unitary matrices that match the Haar distribution up to the second moment. Also, we consider observables O being Pauli strings, so that $\mathrm{Tr}[O] = 0$ and $\mathrm{Tr}\big[O^2\big] = 2^n$. In order to distinguish from the previous notation where averages were computed over the Gaussian distribution of the perturbations, we use $\mathbb{E}_U[\cdot]$ and $\mathrm{Var}_U[\cdot]$ to denote average values and variances evaluated over the random unitaries.

Then, under reasonable and usual assumptions on parts of the parametrized quantum circuit being 2-designs, it is possible to show that the diagonal elements of the Hessian $H_{ii} = \partial^2 f(\boldsymbol{\theta})/\partial\theta_i^2$ satisfy [51] (see also Appendix 8 for an explicit derivation)

$$\mathbb{E}_U[H_{ii}] = 0\,, \quad \mathrm{Var}_U[H_{ii}] \in \mathcal{O}\left(\frac{1}{2^n}\right). \tag{7.16}$$

That is, in addition to first order derivatives, also second order derivatives of random parameterized quantum circuits are found to be zero on average, and with a variance which is exponentially vanishing.

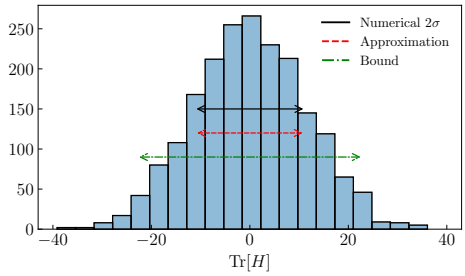


Figure 7.6: Simulation results of evaluating the trace of the Hessian matrix for the circuit shown in Fig. 7.2(b) with random assignments of the parameters and $O = Z^{\otimes 4}$. The simulations are performed by sampling 2000 random parameter vectors $\{\boldsymbol{\theta}_m\}_{m=1}^{2000}$ with $\theta_i \sim \text{Unif}[0, 2\pi[$ and then evaluating the trace of the corresponding Hessian matrix $\text{Tr}[H(\boldsymbol{\theta}_m)]$. These values are used to build the histogram showing the frequency distribution of $\text{Tr}[H]$. The length of the arrows are, respectively: "Numerical 2σ" (black solid line) twice the numerical standard deviation, "Approximation" (dashed red) twice the square root of the approximation in Eq. (7.18), "Bound" (dashed-dotted green) twice the square root of the upper bound in Eq. (7.17).

Starting from the results above, one can calculate the statistics of the trace of the Hessian, for which it holds

$$\mathbb{E}_U[\text{Tr}[H]] = 0\,, \quad \text{Var}_U[\text{Tr}[H]] \lessapprox \frac{M^2}{2^n}\,. \tag{7.17}$$

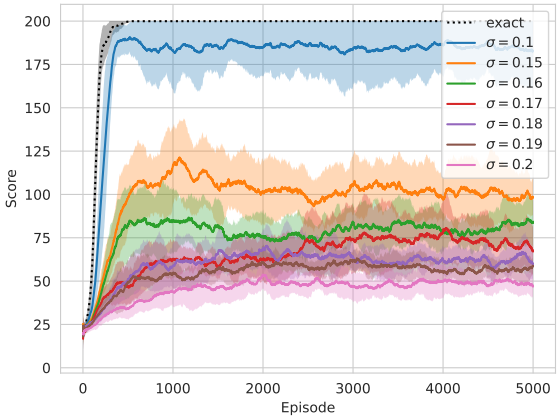
Furthermore, our numerical simulations suggest that the variance of the trace of the Hessian is actually smaller, and is well captured by the following expression

$$\text{Var}_U[\text{Tr}[H]] \approx \frac{M(M+1)}{4(2^n+1)} \approx \frac{1}{4}\frac{M^2}{2^n}\,, \tag{7.18}$$

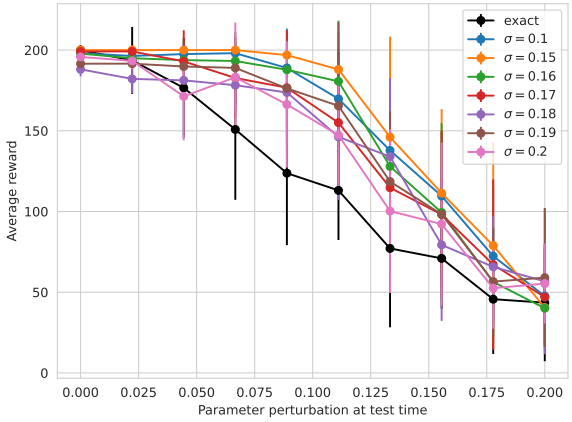
a fact which we justify and discuss in Chapter 8.

In Figure 7.6 we report simulation results of evaluating the trace of the Hessian matrix for the circuit shown in Figure 7.2(b). The histogram represents the frequency of obtaining a given value of the trace of the Hessian $\text{Tr}[H(\boldsymbol{\theta})]$ upon random assignments of the parameters. Indeed, there is a very good agreement between the variance obtained via numerical simulations (black solid line), and the one calculated with the approximation (7.18) (dashed red line).

The circuit used has $M = 92$ parameters and $n = 4$ qubits, and plugging these values in Equation (7.18) yields a standard deviation $\sigma_U = \text{Std}_U[\text{Tr}[H]] \approx 11$. Then, if the behaviour of the PQCs in practical scenarios is well described by its

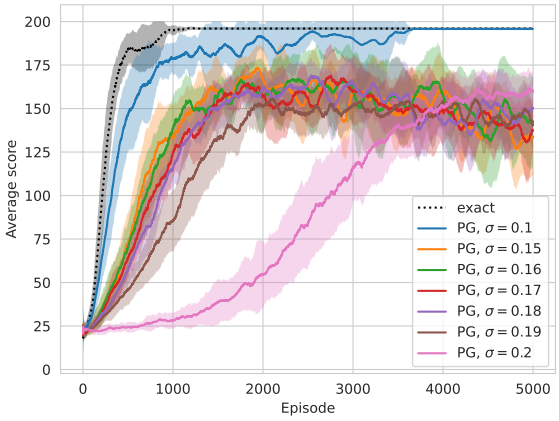


(a) training performance

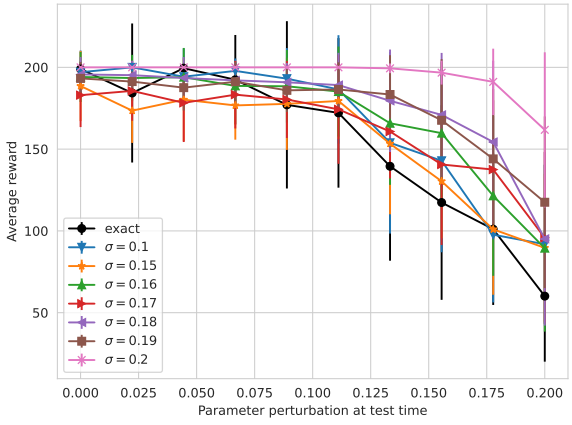


(b) evaluation performance

Figure 7.7: Q-learning agents on the CartPole environment trained and evaluated at varying perturbations σ. Panel (a) shows training performance, while panel (b) shows the performance of the same agents after training and evaluated under different perturbation levels than those present during training. Each point is computed as the average score of the 10 agents under the perturbation indicated on the x-axis.



(a) training performance



(b) evaluation performance

Figure 7.8: Policy gradient agents on the CartPole environment trained and evaluated at varying perturbations σ. Panel (a) shows training performance, while panel (b) shows the performance of the same agents after training and evaluated under different perturbation levels than those present during training. Each point is computed as the average score of the 10 agents under the perturbation indicated on the x-axis.

random parameter regime, one expects the trace of the Hessian to be on average zero and in general not much bigger (in absolute value) than $\sigma_U \approx 11$. With this order of magnitude for the trace, the first order correction Equation (7.15) even with a Gaussian noise level of $\sigma = 0.1$ is very small, as it amounts to

$$|\mathbb{E}[f(\boldsymbol{\theta} + \delta\boldsymbol{\theta})] - f(\boldsymbol{\theta})| \approx \frac{1}{2}\sigma^2 |\mathrm{Tr}[H(\boldsymbol{\theta})]| \approx 0.05\,.$$

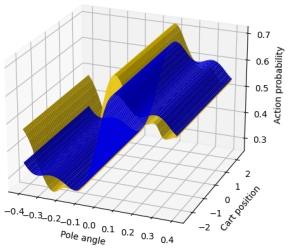
Summing up, for those PQCs whose cost landscape is close to being flat, then Gaussian perturbations on the variational parameters will have a limited impact on the output of the quantum circuit.

7.3.3 Numerical results

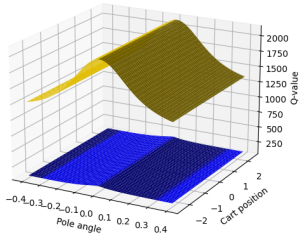
7.3.3.1 CartPole

First, we evaluate the performance of policy gradient and Q-learning algorithms when Gaussian perturbations are applied at each circuit evaluation during training. In Figure 7.7 (a) and (b), we show the training and evaluation performance, respectively, of Q-learning agents in the CartPole environment with perturbations in the range $\sigma \in \{0, 0.1, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2\}$. Only the agent trained with noise level $\sigma = 0.1$ learns the environment successfully and remains close to optimal performance. As suggested by our theoretical analysis in Section 7.3.1, performance starts to degrade as we consider higher perturbations of $\sigma > 0.1$, and none of those agents manage to achieve a better performance than a score of 125 on average. In Figure 7.7 (b) we evaluate the performance of trained agents when they act in an environment with different perturbation levels than those present when they were trained. Even agents that do not perform well during training achieve close to optimal performance when evaluated in the noise-free setting. This suggests that despite their bad training performance due to the added perturbations, these agents still learn a good Q-function. Notably, the agents trained without noise perform worst when they are evaluated under various levels of perturbations.

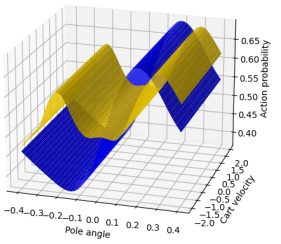
Results for agents trained with the policy gradient method are shown in Figure 7.8 (a). While again only the agents trained with a perturbation of $\sigma = 0.1$ perform well and even reach optimal performance, agents with higher perturbations also largely stay close to optimal performance with a final score of 125 on average. Even the agent trained with a relatively high $\sigma = 0.2$ is robust in this setting, even



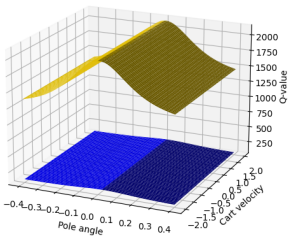
(a) pole angle, cart position (PG)



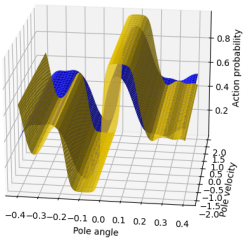
(b) pole angle, cart position (QL)



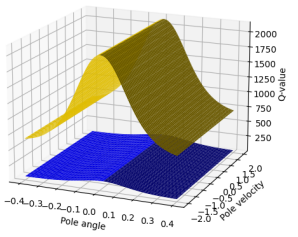
(c) pole angle, cart velocity (PG)



(d) pole angle, cart velocity (QL)



(e) pole angle, pole velocity (PG)



(f) pole angle, pole velocity (QL)

Figure 7.9: Comparison of average learned policies (PG) and Q-functions (QL) of agents from Figure 7.7 and Figure 7.8, in the noise-free setting (blue) and with a perturbation level $\sigma = 0.2$ (yellow).

though it requires by far the most training episodes to get to a good score. This positive trend is also visible in Figure 7.8(b), where we see that all agents achieve close to optimal performance when evaluated with perturbation levels $\sigma \leq 0.1$, which is again in line we our theoretical analysis in section 7.3.1. The difference between agents trained with Gaussian perturbations and those trained without is not as large as in the Q-learning setting, and at evaluation time both algorithms perform similarly. Another observation about the policy gradient agents is that those trained with $\sigma = 0.2$ achieve optimal or close to optimal performance in the environment under various perturbation levels at evaluation time, and are the most robust out of all agents trained in this setting. Overall, the policy gradient method shows a larger resilience to Gaussian noise in our experiments for the CartPole environment. It is an open question why this is the case, however, we did not observe better performance of the policy gradient algorithm under noise in general, as results in later sections will show.

In addition to studying the performance of Q-learning and policy gradient agents at training and evaluation time, we visualize the learned policies and Q-functions of both in the noisy and noise-free setting in Figure 7.9. As learned policies and Q-functions can look different even when training the same agent twice, we show averages of the ten agents shown in Figure 7.7 and Figure 7.8 for both algorithms, and for perturbation levels of $\sigma = 0$ (blue) and $\sigma = 0.2$ (yellow), respectively. The CartPole environment has four inputs: cart position and velocity, and pole angle and velocity. To visualize the learned policies and Q-functions, we show the probabilities and Q-values for taking the action "right" as a function of pairs of state values. The state inputs that are not in the figure are set to zero, and for the sake of clarity we do not apply perturbations to the parameters when visualizing the policy. In Figure 7.9 (a), (c), and (e), we see results for policy gradient agents. Overall, it can be seen that the agents trained without perturbations learn smoother policies, hence for most states there is a clear decision on which action to take. Training with perturbations makes the policies slightly more rippled, but they still mostly follow the contours of the policy learned under ideal conditions.

The approximated Q-functions can be seen in Figure 7.9 (b), (d), and (f). One observation we make here is that the range that Q-values take blows up considerably compared to the noise-free setting. This is due to the trainable output weights that the expectation values are multiplied with in the Q-learning setting (see Section 7.1) becoming considerably larger for agents trained in the noisy setting. However, as

we can see in the Appendix in Figure 7, the shapes of the learned Q-functions of the noise-free and noisy agents are still very similar, which explains why even the agents trained with $\sigma = 0.2$ perform almost optimally when evaluated without perturbations in Figure 7.7 (b). We also note that the range of Q-values of both the noisy and noise-free agents is much larger than the range of optimal Q-values given in [75]. This can be understood as the agent consistently overestimating the expected return, a problem known to arise in classical Q-learning, and which is exacerbated by noise [290]. However, the authors of [75] also point out that in the function approximation setting, it is more important to learn the order of Q-values for each state (i.e., preserving that the argmax Q-value corresponds to the optimal action) than learning a close representation of the optimal Q-values.

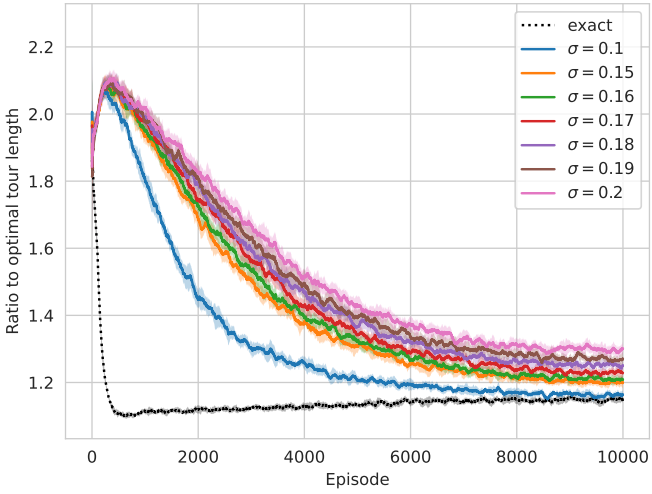
7.3.3.2 Traveling Salesperson Problem

In this section, we study the performance of Q-learning and policy gradient algorithms with Gaussian coherent noise in the TSP environment. Panels (a) and (b) in Figure 7.10 show the training and evaluation performance of Q-learning agents in this environment under perturbations in the range

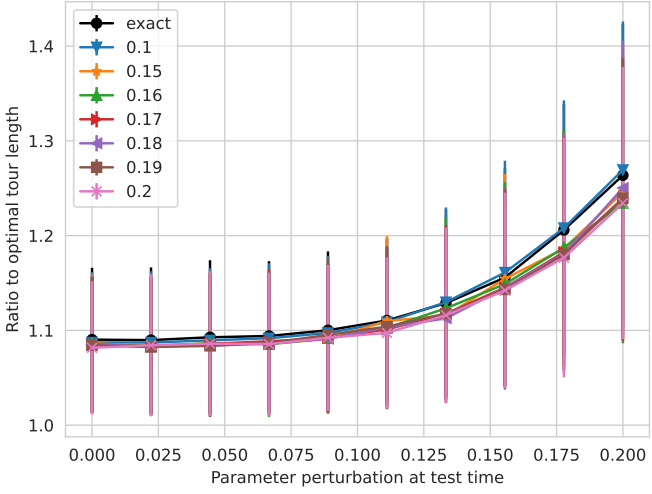
$$\sigma \in \{0, 0.1, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2\}.$$

We note that the Q-learning agents trained without noise already converge after 600 episodes on average, but to get an equal runtime in terms of episodes for all settings, we also let them run for 10000 episodes. This unnecessarily long runtime causes the optimizer to leave the local minimum again, which we ignore as an artifact here and consider the lowest average approximation ratio for the comparison with the other models.

For the TSP environment, we observe that with increasing levels of Gaussian perturbations, convergence of agents is delayed and their final approximation ratio becomes worse compared to the noise-free agents' performance. Still, all agents seem to learn very similar policies despite being trained with different settings of σ, as we can see by their almost identical performance at evaluation time shown in Figure 7.10 (b). Despite a drop in performance during training, the final performance of the models on a test set of previously unseen TSP instances stays almost unaffected by the noise present during training. While we see that agents trained with more noise seem to learn more noise-robust policies as in the case of the CartPole environment, this effect is not as pronounced here. Additionally,

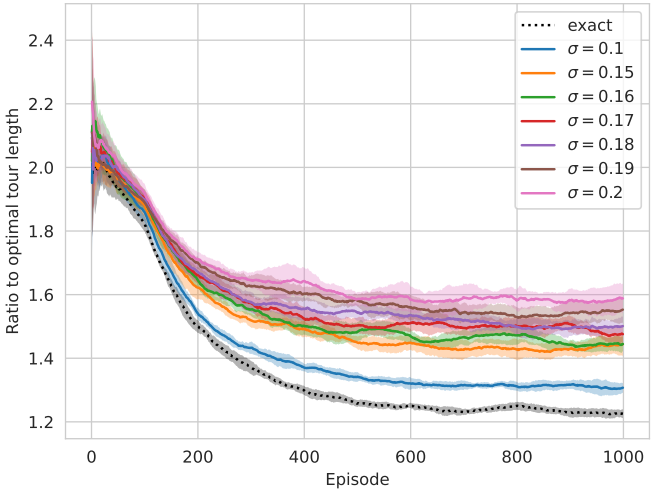


(a) training performance

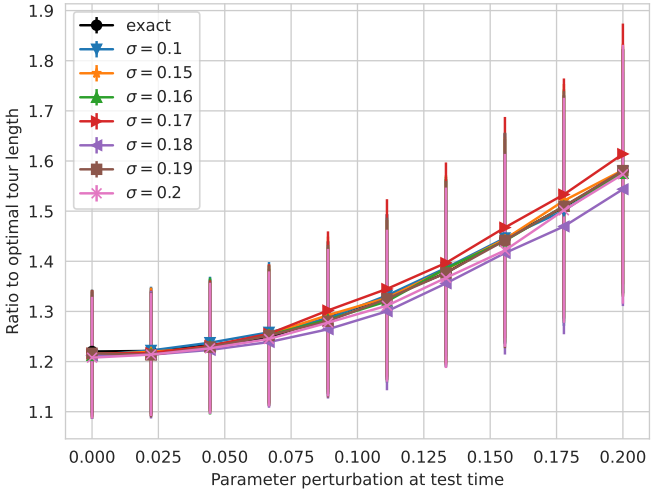


(b) evaluation performance

Figure 7.10: Training and evaluation of Q-learning agents in the TSP environment under various perturbations σ. Panel (a) shows the effect of perturbations during training, panel (b) shows results for the same agents evaluated on varying perturbation levels after training, different to those present at training time.



(a) training performance



(b) evaluation performance

Figure 7.11: Training and evaluation of policy gradient agents in the TSP environment under various perturbations σ. Panel (a) shows the effect of perturbations during training, panel (b) shows results for the same agents evaluated on varying perturbation levels after training, different to those present at training time.

we again see that performance of trained models in Figure 7.10 (b) starts to drop at $\sigma > 0.1$, as indicated by our theoretical analysis in Section 7.3.1. While the policy gradient method shows a certain robustness to noise during training in the CartPole environment, this is not the case for the TSP environment, as we show in Figure 7.11 (a). The only agent that gets close in performance to the noise free agent is the one trained with $\sigma = 0.1$, while higher perturbations yield agents that are relatively bad with an approximation ratio between 1.4 and 1.6 on average. However, again, all agents seem to learn similar policies as indicated by their test performance in Figure 7.11 (b). Similar to CartPole, the agents' performance on the test set under varying perturbation levels closely matches that of the noise-free agents, and again we see a large drop in performance for perturbations that are higher than $\sigma = 0.1$.

Overall, the Q-learning algorithm performs better in the TSP environment than the policy gradient method. The optimal tour for each TSP instance is deterministic, so using a stochastic policy as in the policy gradient approach introduces an additional source of error, as there is always a non-zero probability to chose a non-optimal action. This leads to an increased susceptibility to the Gaussian perturbations present during the evaluation of the policy gradient algorithm. This is not the case for Q-learning, where choices are made based on the argmax Q-value. Additionally, the ansatz that we use does not separate between data encoding and trainable parameters as described in Section 7.1. As the optimal tour of a TSP instance does not change upon small perturbations of the edge weights, this leads to a relative robustness of this ansatz used in conjunction with Q-learning to Gaussian coherent noise in this environment.

7.4 Incoherent noise

The Gaussian perturbation noise that we studied in Section 7.3 is well-suited to model coherent errors due to imprecision in the control of the quantum device, but it does not reflect noise that results from undesired interactions of the quantum system with its environment. To study the effect of this type of incoherent noise we perform additional experiments in this section.

We simulate this type of noise with TensorFlow Quantum (TFQ) [224], where they are implemented through a Monte-Carlo trajectory sampling method [291, 292] that approximates the effect of noise by averaging over state vectors generated from

a probabilistic application of the noise channel. This method of simulating noise essentially trades off the overhead in memory needed to store the $2^n \times 2^n$ sized density matrices necessary to simulate incoherent noise, with a runtime overhead. The precision of this approximation is determined by the number of repetitions, which specifies how many "noisy" state vectors are used. This adds a stochastic element to the simulation of the noise channels, and we get closer to simulating the exact noise model as the number of trajectories increases. Depending on the environments, we choose the number of trajectories so that it is possible to perform simulations in a reasonable time frame, and specify this number individually for each of the experiments below. We note that the runtime requirements for CartPole when simulating this type of noise are especially high, as the number of time steps in each episode, as well as the number of episodes itself depends strongly on the performance of the agent. In particular, agents that perform neither very well nor very poorly, which are exactly the noise configurations we are interested in studying here, take especially long to simulate, as they do not converge early by solving the environment, but still take on the order of 100 time steps in each episode. Therefore we focus our attention mainly on the TSP environment in this section.

7.4.1 Depolarizing noise

Depolarizing noise affects a quantum state by either replacing it with the completely mixed state with probability p, or leaving it untouched otherwise [293]. Let ρ be the density matrix of a qubit, then depolarizing noise is defined by the map

$$\mathcal{D}_p(\rho) = (1-p)\rho + p\frac{\mathbb{1}}{2}. \tag{7.19}$$

We model depolarization noise with Cirq [291] and TFQ [224] by appending a layer of local depolarizing channels to every qubit after each time step of the computation, where a time step is defined as the largest set of gates that can be implemented simultaneously. This implementation takes into account the possibility of cross-talk between qubits [294]. Also, note that while the use of depolarizing channels alone may not be a good approximation of real single qubits errors, it may become a good effective description of the overall noise process for the case where many qubits and layers are used [295].

In our simulations, we assume that both single- and two-qubits gates are noisy, and consist of a composition of the ideal gates followed by *local* depolarizing channels

of equal probability p, acting independently on each qubit. In particular, the application of a depolarizing noise channel is implemented by performing one out of four actions at each circuit execution (trajectory): do nothing with probability $1-p$, or apply at random one of the three Pauli operators with probability p, and then average over the results. We remark that the average gate error of single-qubit gates in currently available superconducting quantum computing hardware is of the order of $r \lessapprox 0.01$, with gate fidelities exceeding $> 99\%$. Finally, we note that one can relate the depolarisation strength p to the average gate error r over single qubit Cliffords, as measured by Randomized Benchmarking (RB) [296, 294, 296] and commonly reported for quantum devices [297, 298], via $r = p/2$. However, our circuits do not only use Cliffords, and moreover, estimates for the gate error in RB depend on the basis gates available on the device. Therefore, one should consider our simulations with depolarizing noise of strength p as a proxy for a quantum device whose average error rate r is of the same order of magnitude of p. While a single-qubit error noise model may not be accurate enough to closely mimic the behaviour of a real quantum device, it gives us the possibility to study the effect of single-qubit errors separately, before we go on to study a noise model that also includes two-qubit gate errors in section 7.4.2.

As mentioned above, simulating incoherent noise has high runtime requirements, so in the following we limit our studies to: (i) Q-learning in the CartPole environment, and (ii) the policy gradient method in the TSP environment. We pick these settings as they were the ones that were more sensitive to Gaussian coherent noise in our studies in Section 7.3, and in that sense represent the worst case instances from the previous section. To simulate the noisy quantum circuits, we use the Monte Carlo sampling as described above, where the number of trajectories used depends on the environment. As the CartPole environment requires a very high number of environment interactions (the better the agent, the more circuit evaluations are required per episode), we use 100 trajectories in this setting. In the TSP environment, the number of steps in each episode is constant and therefore we can use a higher number of 1000 trajectories and still perform simulations in a timely manner.

Figure 7.12 shows results of Q-learning agents trained in the CartPole environment with various error probabilities p. Agents with a realistic error probability of up to $p = 0.01$ still solve the environment in less than 2000 episodes on average. Agents trained with error probability $p = 0.005$ reach higher scores almost as quickly as

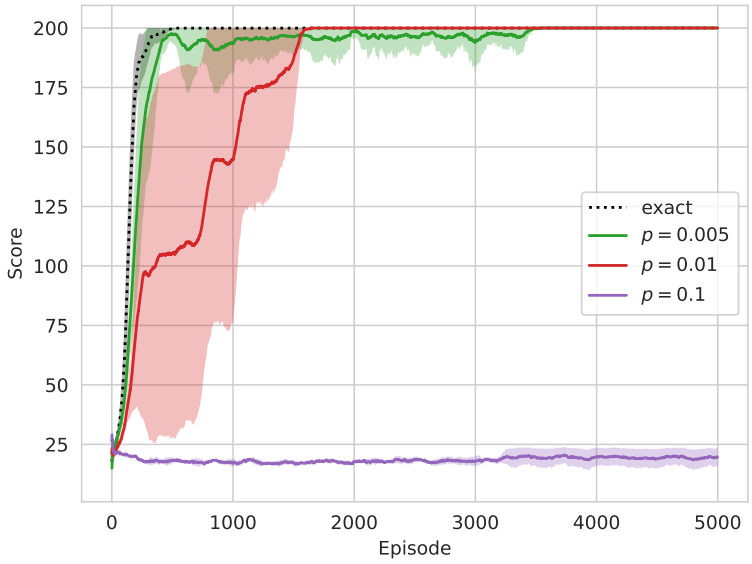


Figure 7.12: Q-learning agents trained with varying probabilities p of depolarization errors, and five layers of the circuit depicted in Figure 7.2 a). Noise is simulated with 100 Monte Carlo trajectories. The noisy curves are averaged over 5 agents, the exact one is averaged over 10 agents as in previous figures.

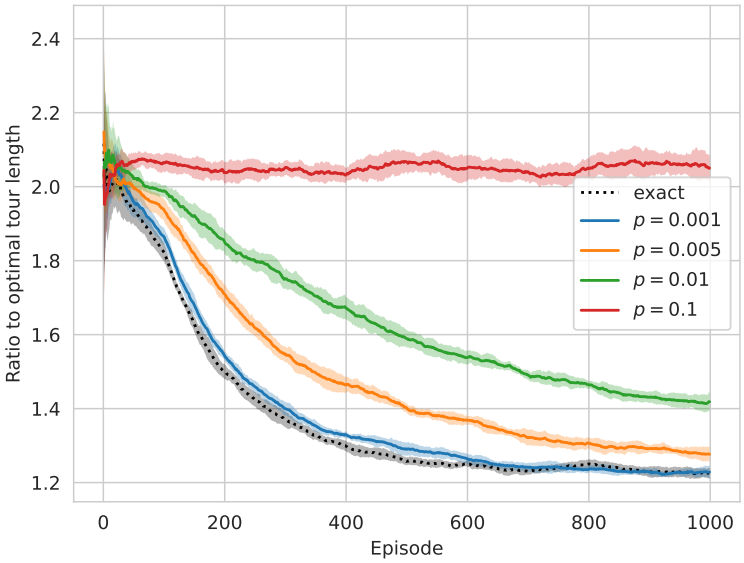


Figure 7.13: Policy gradient agents trained in the TSP environment with varying probabilities p of depolarization error, with one layer of the circuit depicted in Figure 7.2 c). Noise is simulated with 1000 Monte Carlo trajectories. All curves are averaged over 10 agents.

agents trained in the noise-free setting, but stay somewhat unstable until they solve the environment after 3500 episodes on average. When the noise probability is increased to $p = 0.1$, we see that agents fail to make any learning progress at all.

Figure 7.13 shows the performance of the policy gradient method under one-qubit depolarization errors in the TSP environment. In this setting, agents trained with error probability $p = 0.01$, as is a realistic assumption on current devices, perform noticeably worse than agents in the noise-free setting with a drop in approximation ration of around 0.2 on average. Only when we consider an error probability of $p = 0.001$ do we get performance that is almost exactly the same as that in the noise-free case. Similar to the results of the Q-learning agent in the CartPole environment, agents trained with an error probability of $p = 0.1$ show no meaningful learning progress.

7.4.2 Noise model based on current hardware

After studying the effect of single-qubit depolarization errors in Section 7.4.1, we now study the performance of the Q-learning algorithm in the TSP environment in the presence of a more realistic noise model that captures the behaviour of a near-term superconductive quantum device. The error sources we incorporate into this noise model are the following: single-qubit and two-qubit depolarization errors, single qubit amplitude damping error, and measurement noise. While hardware providers like IBM and Google offer the possibility of simulating noise models of specific devices, we do not want to take device-specific factors like qubit topology and native gate sets into account in this work, as the performance in these settings also depends strongly on the quality of the circuit compiled to the native gate set and qubit connectivity [299]. Instead, we define a custom noise model based on gate fidelities published by hardware vendors, but do not take the above details into account. To determine realistic settings for the error probability of each noise source, we use calibration data published by IBM [300] at the time of writing. The noise model used in our simulation is specified as follows:

- **Depolarization error:** Single qubit depolarization channels with $p = 0.001$ are applied after every single qubit gate. Two-qubit depolarization errors, defined by properly adjusting the definition in Equation (7.19), with $p_2 = 0.01$ are applied after every two-qubit gate on the corresponding pair of qubits.

Error source	**a)**	b)	c)	d)
Depolarization (1Q)	**0.001**	0.001	0.01	0.1
Depolarization (2Q)	**0.01**	0.01	0.1	0.2
Amplitude damping	**0.0003**	0.03	0.03	0.1
Bitflip (measurement)	**0.01**	0.01	0.1	0.1

Table 7.1: Error strengths for the configurations of the custom noise model used in Figure 7.14. Depolarization (1Q) indicates the single qubit depolarising channel applied after each single-qubit gate, and similarly for 2Q for two-qubit gates. Configuration a) in bold is based on error rates published by IBM at the time of writing, as described in the main text.

- **Amplitude damping error:** Amplitude damping channels with decay parameter $\gamma = 0.003$ are applied after each single- and two-qubit gate on the corresponding qubits. Such a decay rate is valid for real devices having single qubit gate durations of $t = 35$ns, and average qubit decay times $T_1 \approx 100\mu$s, which correspond to a decay parameter of $\gamma = 1 - \exp(-t/T1) \approx 0.0003$.
- **Measurement noise** Measurement errors are modeled by appending a bit-flip channel with probability $p = 0.01$ to every qubit right before the measurement process.

We recall that the circuit ansatz for the TSP environment is the one depicted in Figure 7.2(c), where input information about the edge weights of the TSP instance is encoded by means of two-qubit gates. We therefore chose to study this ansatz in the context of a noise model that incorporates two-qubit errors, as we expect that these types of errors will affect performance of an ansatz that encodes crucial information in two-qubit gates more severely. Additionally, it is hard to perform simulations in this setting for the CartPole environment in a reasonable amount of time, as discussed above. For these reasons, we restrict our attention to the TSP environment in this section.

Figure 7.14 shows results averaged over five Q-learning agents in the TSP environment for each of the error probability configurations of the custom noise model described above. We show the specific error probabilities used for the simulations in Table 7.1. Configuration a) corresponds to error probabilities that are consistent with those present on current quantum hardware as described above. Based on this, we specify three other error probabilities b) - d) by increasing the error on varying error sources. We note that while the error probabilities themselves in

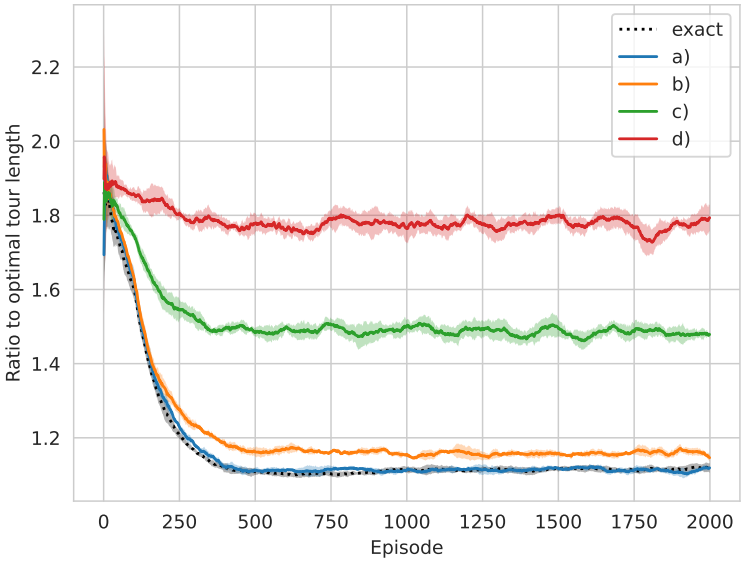


Figure 7.14: Q-learning agents trained in the TSP environment with one layer of the circuit depicted in Figure 7.2 c) and custom noise model, using 1000 Monte Carlo trajectories. The labels indicate the custom noise configurations defined in Table 7.1, results are averaged over five agents in each curve, except for the exact curve which is averaged over ten agents as done in previous figures.

configuration a) are consistent with those on current hardware, our simulation is only an approximation of this error due to the Monte Carlo trajectory sampling method described in Section 7.4. To perform simulations in a reasonable time frame, we use 1000 trajectories for each circuit evaluation. The circuit that we simulate has 145 gates (counting a ZZ-gate as two CNOTs and one Z gate), and for small error probabilities the chance of applying each of the noise channels is relatively small. This means that in each trajectory, a relatively small number of noise channels is applied. Hence we expect that the results in Figure 7.14 are slightly better than what we would get if the exact noise model was simulated (i.e., in the limit of a large number of trajectories, or by considering the full density matrix).

Looking at the results in Figure 7.14, we see that for configuration a) (blue), the performance of the agents matches those of the noise-free ones (dotted black)

almost exactly, and the noise model based on realistic error strengths of current devices does not affect training. We see a slight drop in performance when we increase the error probability of the amplitude damping channels from 0.0003 to 0.03 (orange), as described in Table 7.1, column b). For configuration c), we also increase the other remaining error sources' probabilities, which leads to a considerable drop in performance. In configuration d), we assume extremely high error probabilities for each of the noise channels, which leads to a complete failure of the agents to make any meaningful learning progress in this environment.

7.5 Conclusions

Our goal in this chapter was to evaluate the resilience of variational RL algorithms to various types of noise that are present on real quantum hardware. First, we investigated shot noise, which results from the probabilistic nature of quantum measurements. We introduced a method to reduce the number of shots to train a Q-learning agent, motivated by the specific structure of the underlying RL algorithm. Our shot allocation technique enables a more shot-frugal training of variational Q-learning models with little or no effect on the final performance of the agents.

After considering shot noise, we moved on to study the effect of Gaussian coherent errors that can arise on real hardware due to miscalibration of the device, or imprecise pulse sequences that implement the parameterised gates in the quantum circuit. We gave an analytic expression for how this type of noise affects the output of a quantum RL agent, and provided a bound on the standard deviation of the Gaussian error that elucidates the tolerable magnitude of the error on the output of a quantum model. We confirm this bound in our simulations, where we study the effect of various levels of Gaussian perturbations on the performance of training policy gradient and Q-learning agents in two different environments. For one of these environments, we find that agents trained with higher noise probabilities also learn more robust policies and Q-functions, in the sense that under evaluation of different perturbation levels, these agents achieve optimal or close to optimal performance more often.

Finally, we studied incoherent noise that emerges in real hardware due to undesired interactions of the qubits with the surrounding environment, as the device is not completely shielded from external effects. To this end, we consider single-qubit

depolarization errors, as well as a custom noise model that combines single- and two qubit depolarization errors, amplitude damping errors, and bitflip (measurement) errors. For the latter, we perform simulations with realistic error probabilities for each of the noise channels, in line with data published for IBM devices at the time of writing.

Overall, we find that the effect of noise on training variational RL algorithms for Q-learning and the policy gradient method depends strongly on the strength of the noise, as well as the type of noise itself. For some cases, like decoherence errors with realistic error probabilities of current devices, the drop in performance is relatively small. On the other hand, we find that large Gaussian perturbations as well as errors induced by the probabilistic nature of quantum measurements can affect performance in highly detrimental ways. Additionally, we find that for Gaussian coherent noise agents that are trained with higher perturbations learn more noise-robust policies in some cases, similar to results in classical literature, where noise is used as a regularization technique.

While our results were performed in a regime that is still efficiently simulable on classical computers, it is an interesting question for future work to consider the implications of noise-robustness of large-scale quantum models in light of recent results which show that in certain settings, the outputs of noisy quantum circuits can be efficiently approximated classically [50, 301]. This raises the question to what extent an inherent noise-robustness of hybrid variational quantum machine learning affects the possibility to achieve a quantum advantage with these types of models.

On the practical side, the optimization procedures that we used in this work were the same as those commonly used to train models in noise-free simulations and are not tailored to account for quantum hardware specific noise. This raises the question on how optimization methods that are tailored for the special characteristics of variational quantum models could further improve the performance of these types of models in a noisy setting. For the optimization of PQC parameters in the combinatorial optimization or quantum chemistry setting, it is known that some optmization methods, like simultaneous perturbation stochastic approximation (SPSA), actually become better with noise. It is an interesting area of future research to design quantum-specific optimization routines for machine learning that address or even combat specific types of noise, for example leveraging effective quantum error mitigation techniques [302, 303, 304]. This work motivates the study

of these types of optimization methods, as well as continued efforts to find learning tasks where variational RL algorithms can potentially provide an advantage.