# Algorithm selection and configuration for Noisy Intermediate Scale Quantum methods for industrial applications

Moussa, C.

# Chapter 7

# Hyperparameter Importance of Quantum Neural Networks Across Small Datasets

Machine learning has not been spared in the quest for meaningful quantum computing applications. One of the more investigated approaches is the use of a special type of quantum circuit – a so-called quantum neural network – to serve as a basis for a machine learning model. We introduced the latter workflow in Chapter 2.5. Roughly speaking, as the name suggests, a quantum neural network can play a similar role to a neural network. However, specifically for applications in machine learning contexts, very little is known about suitable circuit architectures, or model hyperparameters one should use to achieve good learning performance. In this chapter[1], we apply the functional ANOVA framework to quantum neural networks to analyze which of the hyperparameters were most influential for their predictive performance. We analyze one of the most typically used quantum neural network architectures.

[1]Contents of this chapter are based on [137]; Charles Moussa, Jan N. van Rijn, Thomas Bäck, and Vedran Dunjko. Hyperparameter importance of quantum neural networks across small datasets. In Poncelet Pascal and Dino Ienco, editors, *Discovery Science*, pages 32–46, Cham, 2022. Springer Nature Switzerland.

## 7.1 Introduction

Quantum models can exhibit clear potential in special datasets where we have theoretically provable separations with classical models [94, 117, 163, 188]. More theoretical works also study these models from a generalization perspective [38]. Quantum circuits with adjustable parameters, also called quantum neural networks, have been used to tackle regression [130], classification [83], generative adversarial learning [208], and reinforcement learning tasks [94, 180].

However, the value of quantum machine learning on real-world datasets is still to be investigated in any larger-scale systematic fashion [82, 154]. Currently, common practices from machine learning, such as large-scale benchmarking, hyperparameter importance, and analysis have been challenging tools to use in the quantum community [171]. Given that there exist many ways to design quantum circuits for machine learning tasks, this gives rise to a hyperparameter optimization problem. However, there is currently limited intuition as to which hyperparameters are important to optimize and which are not. Such insights can lead to much more efficient hyperparameter optimization [31, 60, 133].

In order to fill this gap, we employ functional ANOVA [92, 181], a tool for assessing hyperparameter importance. This follows the methodology of [192, 175], who employed this across datasets, allowing for more general results. For this, we selected a subset of several low-dimensional datasets from the OpenML-CC18 benchmark [22], that are matching the current scale of simulations of quantum hardware. We defined a configuration space consisting of ten hyperparameters from an aggregation of quantum computing literature and software. We extend this methodology by an important additional verification step, where we verify the performance of the internal surrogate models. Finally, we perform an extensive experiment to verify whether our conclusions hold in practice. While our main findings are in line with previous intuition on a few hyperparameters and the verification experiments, we also discovered new insights. For instance, setting well the learning rate is deemed the most critical hyperparameter in terms of marginal contribution on all datasets, whereas the particular choice of entangling gates used is considered the least important except on one dataset.

## 7.2 Background

In this section, we introduce the necessary background on functional ANOVA, quantum computing, and quantum circuits with adjustable parameters for supervised learn-

ing.

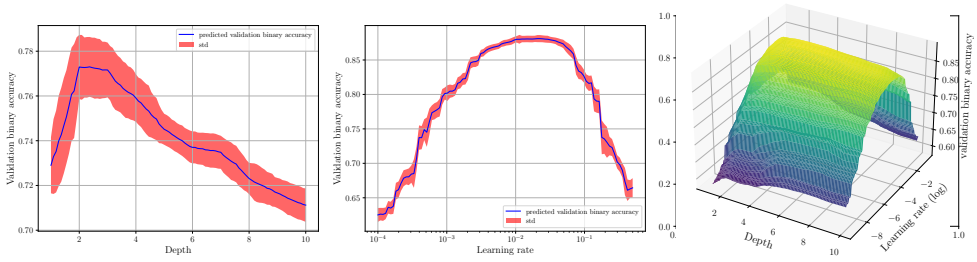
### 7.2.1 Functional ANOVA

When applying a new machine learning algorithm, it is unknown which hyperparameters to modify in order to get high performances on a task. Several techniques exist that assess hyperparameter importance, such as functional ANOVA [164]. The latter framework can detect the importance of both individual hyperparameters and interaction effects between different subsets of hyperparameters. We first introduce the relevant notation, based on the work by Hutter *et al.* [92].

Let $A$ be a machine learning algorithm that has $n$ hyperparameters with domains $\Theta_1, \ldots, \Theta_n$ and *configuration space* $\boldsymbol{\Theta} = \Theta_1 \times \ldots \times \Theta_n$. An instantiation of $A$ is a vector $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_n\}$ with $\theta_i \in \Theta_i$ (this is also called a *configuration* of $A$). A partial instantiation of $A$ is a vector $\boldsymbol{\theta}_U = \{\theta_{i_1}, \ldots, \theta_{i_k}\}$ with a subset $U = \{i_1, \ldots, i_k\} \subseteq N = [n] = \{1, \ldots, n\}$ of the hyperparameters fixed, and the values for other hyperparameters unspecified. Note that $\boldsymbol{\theta}_N = \boldsymbol{\theta}$.

Functional ANOVA is based on the concept of a marginal of a hyperparameter, i.e., how a given value for a hyperparameter performs, averaged over all possible combinations of the other hyperparameters' values. The *marginal performance* $\hat{a}_U(\boldsymbol{\theta}_U)$ is described as the average performance of all complete instantiations $\boldsymbol{\theta}$ that have the same values for hyperparameters that are in $\boldsymbol{\theta}_U$. As an illustration, Fig. 7.1 shows marginals for two hyperparameters of a quantum neural network and their union. As the number of terms to consider for the marginal can be very large, the authors of [92] used tree-based surrogate regression models to calculate efficiently the average performance. Such a model yields predictions $\hat{y}$ for the performance $p$ of arbitrary hyperparameter settings.

Functional ANOVA determines how much each hyperparameter (and each combination of hyperparameters) contributes to the variance of $\hat{y}$ across the algorithm's hyperparameter space $\boldsymbol{\Theta}$, denoted $\mathbb{V}$. Intuitively, if the marginal has a high variance, the hyperparameter is highly important to the performance measure. Such framework has been used for studying the importance of hyperparameters of common machine learning models such as support vector machines, random forests, Adaboost, and residual neural networks [192, 175]. We refer to [92] for a complete description and introduce the quantum supervised models considered in this study along with the basics of quantum computing.




**(a)** **(b)** **(c)**

Figure 7.1: Examples of marginals for a quantum neural network with validation accuracy as performance on the banknote-authentication dataset. The hyperparameters correspond to the learning rate used during training (a), the number of layers, also known as depth (b), and their combination (c). The hyperparameter values for the learning rate are on a log scale. When considered individually, we see for instance that depth and learning rate should not be set too high for better performances. However, when grouped together, the learning rate seems most influential.

### 7.2.2 Supervised learning with Parameterized Quantum Circuits

A parameterized quantum circuit (also called *ansatz*) can be represented by a quantum circuit with adjustable real-valued parameters $\boldsymbol{\theta}$. The latter is then defined by a unitary $U(\boldsymbol{\theta})$ that acts on a fixed $n$-qubit state (e.g., $|0^{\otimes n}\rangle$). The ansatz may be constructed problem-independent generic construction. The latter are often designated as *hardware-efficient*.

For a machine learning task, this unitary encodes an input data instance $x \in \mathbb{R}^d$ and is parameterized by a trainable vector $\boldsymbol{\theta}$. Many designs exist but hardware-efficient parameterized quantum circuits [97] with an alternating-layered architecture are often considered in quantum machine learning when no information on the structure of the data is provided. This architecture is depicted in an example presented in Fig. 7.2 and essentially consists of an alternation of encoding unitaries $U_{\text{enc}}$ and variational unitaries $U_{\text{var}}$. In the example, $U_{\text{enc}}$ is composed of single-qubit rotations $R_X$, and $U_{\text{var}}$ of single-qubit rotations $R_z, R_y$ and entangling Ctrl-$Z$ gates, represented as ┇ in Fig. 7.2, forming the entangling part of the circuit. Such entangling part denoted $U_{\text{ent}}$, can be defined by connectivity between qubits.

These parameterized quantum circuits are similar to neural networks where the circuit architecture is fixed and the gate parameters are adjusted by a classical optimizer such as gradient descent. They have also been named quantum neural networks.

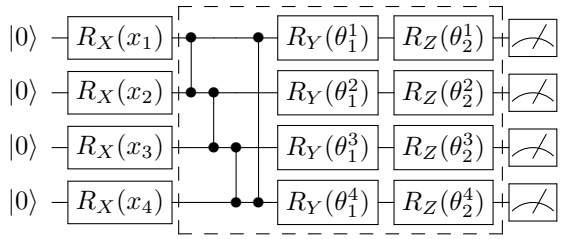



Figure 7.2: Parameterized quantum circuit architecture example with 4 qubits and ring connectivity (qubit 1 is connected to 2, 2 to 3, 3 to 4, and 4 to 1 makes a ring). The first layer of $R_X$ is the encoding layer $U_{\text{enc}}$, taking a data instance $x \in \mathbb{R}^4$ as input. It is followed by the entangling part with Ctrl-$Z$ gates. Finally, a variational layer $U_{\text{var}}$ is applied. Eventually, we do measurements to be converted into predictions for a supervised task. The dashed part can be repeated many times to increase the expressive power of the model.

The parameterized layer can be repeated multiple times, which increases its *expressive power* like neural networks [179]. The data encoding strategy (such as reusing the encoding layer multiple times in the circuit - a strategy called *data reuploading*) also influences the latter [151, 174].

Finally, the user can define the observable(s) and the post-processing method to convert the circuit outputs into a prediction in the case of supervised learning. Commonly, observables based on the single-qubit $Z$ operator are used. When applied on $m \leq n$ qubits, the observable is represented by a $2^m - 1$ square diagonal matrix with $\{-1, 1\}$ values and is denoted $\mathcal{O} = Z \otimes Z \otimes \cdots \otimes Z$.

Having introduced parameterized quantum circuits, we present the hyperparameters of the models, the configuration space, and the experimental setup for our functional ANOVA-based hyperparameter importance study.

## 7.3 Methods

In this section, we describe the network type and its hyperparameters and define the methodology that we follow.

### 7.3.1 Hyperparameters and configuration space

Many designs have been proposed for parameterized quantum circuits depending on the problem at hand or motivated research questions and contributions.

Such propositions can be aggregated and translated into a set of hyperparameters and configuration space for the importance study. As such, we first did an extensive literature review on parameterized quantum circuits for machine learning [18, 83, 84, 93, 94, 116, 123, 129, 130, 154, 168, 180, 194, 195, 202, 208] as well as quantum machine learning software [6, 19, 34]. This resulted in a list of 10 hyperparameters, presented in Table 7.1. We choose them so we balance between having well-known hyperparameters that are expected to be important, and less considered ones in the literature. For instance, many works use Adam [103] as the underlying optimizer, and the learning rate should generally be well chosen. On the contrary, the entangling gate used in the parameterized quantum circuit is generally a fixed choice.

From the literature, we expect data encoding strategy/circuit to be important. We choose two main forms for $U_{\text{enc}}$. The first one is the hardware-efficient $\bigotimes_{i=1}^{n} R_X(x_i)$. The second takes the following form from [19, 93, 83]:

$$U_{\text{enc}}(\boldsymbol{x}) = U_z(\boldsymbol{x}) H^{\otimes n} \tag{7.1}$$

$$U_z(\boldsymbol{x}) = \exp\left( -\mathrm{i}\pi \left[ \sum_{i=1}^{n} x_i Z_i + \sum_{\substack{j=1, \\ j>i}}^{n} x_i x_j Z_i Z_j \right] \right). \tag{7.2}$$

Using data-reuploading [151] results in a more expressive model [174], and this was also demonstrated numerically [94, 151, 180]. Finally, pre-processing of the input is also sometimes used in encoding strategies that directly feed input features into Pauli rotations. It also influences the expressive power of the model [174]. In this work, we choose a usual activation function *tanh* commonly used in neural networks. We do so as its range is $[-1, 1]$, which is the same as the data features during training after the normalization step.

The list of hyperparameters we take into account is non-exhaustive. It can be extended at will, at the cost of more software engineering and budget for running experiments.

### 7.3.2 Assessing Hyperparameter Importance

Once the list of hyperparameters and configuration space are decided, we perform the hyperparameter importance analysis with the functional ANOVA framework. Assessing the importance of the hyperparameters boils down to four steps. Firstly, the models are applied to various datasets by sampling various configurations in a hyperparameter

optimization process. The performances or metrics of the models are recorded along. The sampled configurations and performances serve as data for functional ANOVA. As functional ANOVA uses internally tree-based surrogate models, namely random forests [32], we decided to add an extra step with reference to [192]. In the second step, we verify the performance of the internal surrogate models. We cross-evaluate them using regression metrics commonly used in surrogate benchmarks [53]. Surrogates performing badly at this step are then discarded from the importance analysis, as they can deteriorate the quality of the study. Thirdly, the marginal contribution of each hyperparameter over all datasets can be then obtained and used to infer a ranking of their importance. Finally, a verification step similar to [192] is carried out to confirm the inferred ranking previously obtained. We explain such a procedure in the following section.

### 7.3.3 Verifying Hyperparameter Importance

When applying the functional ANOVA framework, an extra verification step is added to confirm the output from a more intuitive notion of hyperparameter importance [192]. It is based on the assumption that hyperparameters that perform badly when fixed to a certain value (while other hyperparameters are optimized), will be important to optimize. The authors of [192] proposed to carry out a costly random search procedure fixing one hyperparameter at a time. In order to avoid a bias to the chosen value to which this hyperparameter is fixed, several values are chosen, and the optimization procedure is carried out multiple times. Formally, for each hyperparameter $\theta_j$ we measure $y^*_{j,f}$ as the result of a random search for maximizing the metric, fixing $\theta_j$ to a given value $f \in F_j, F_j \subseteq \Theta_j$. For categorical $\theta_j$ with domain $\Theta_j$, $F_j = \Theta_j$ is used. For numeric $\theta_j$, the authors of [192] use a set of 10 values spread uniformly over

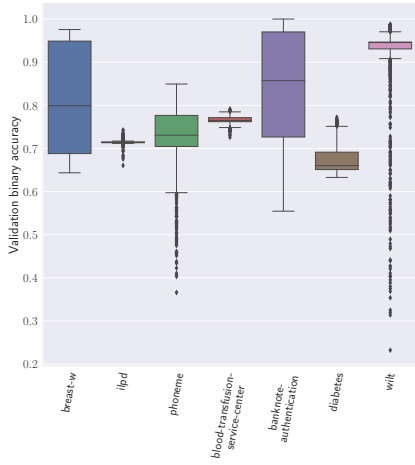



Figure 7.3: Performances of 1 000 quantum machine learning models defined by different configurations of hyperparameters over each dataset. The metric of interest in the study is the 10-fold cross-validation accuracy. We take the best-achieved metric per model trained over 100 epochs.

$\theta_j$'s range. We then compute $y_j^* = \frac{1}{|F_j|}\sum_{f\in F_j} y_{j,f}^*$, representing the score when not optimizing hyperparameter $\theta_j$, averaged over fixing $\theta_j$ to various values it can take. Hyperparameters with lower values for $y_j^*$ are assumed to be more important since the performance should deteriorate more when set sub-optimally.

In our study, we extend this framework to be used on the scale of quantum machine learning models. As quantum simulations can be very expensive, we carry out the verification experiment by using the predictions of the surrogate instead of fitting new quantum models during the verification experiment. The surrogates yield predictions $\hat{y}$ for the performance of arbitrary hyperparameter settings sampled during a random search. Hence, they serve to compute $y_{j,f}^*$. This is also why we assessed the quality of the built-in surrogates as the second step. Poorly-performing surrogates can deteriorate the quality of the constructed marginals, and therefore lead to poorly-supported conclusions.

## 7.4 Dataset and inclusion criteria

To apply our quantum models and study the importance of the previously introduced hyperparameters, we consider classical datasets. Similarly to [192], we use datasets from the OpenML-CC18 benchmark suite [22]. In our study, we consider only the case where the number of qubits available is equal to the number of features, a common setting in the quantum community. As simulating quantum circuits is a costly task, we limit this study to the case where the number of features is less than 20 after preprocessing.[2] Our first step was to identify which datasets fit this criterion. We include all datasets from the OpenML-CC18 that have 20 or fewer features after categorical hyperparameters have been one-hot-encoded, and constant features are removed. Afterwards, the input variables are also scaled to unit variance as a normalization step. The scaling constants are calculated on the training data and applied to the test data.

The final list of datasets is given in Table 7.2. In total, 7 datasets fitted the criterion considered in this study. For all of them, we picked the OpenML Task ID giving the 10-fold cross-validation task. A quantum model is then applied using the latter procedure, with the aforementioned preprocessing steps.

[2] A 10-fold cross-validation run in our experiment takes on average 262 minutes for 100 epochs with Tensorflow Quantum [34].

## 7.5 Results

In this section, we present the results obtained using the hyperparameters and the methodology defined in Section 7.3 with the datasets described in Section 7.4. First, we show the distribution of performances obtained during a random search where configurations are independently sampled for each dataset. Then we carry out the surrogate verification. Finally, we present the functional ANOVA results in terms of hyperparameter importance with marginal contributions and the random search verification per hyperparameter.

### 7.5.1 Performance distributions per dataset

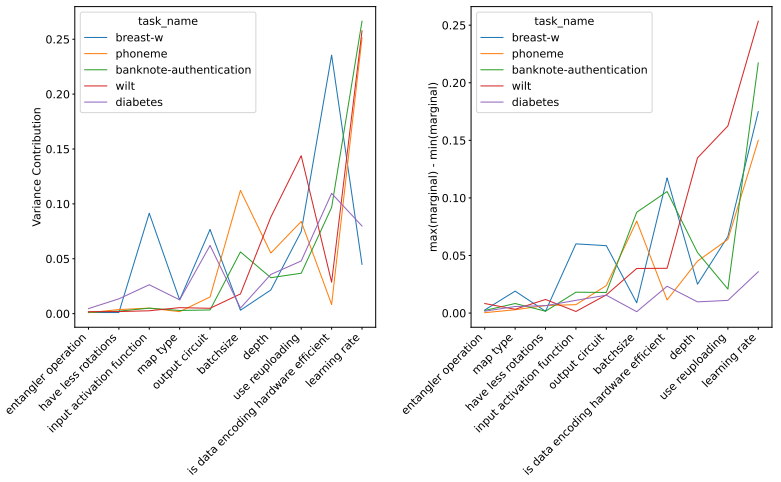
For each dataset, we sampled independently 1 000 hyperparameter configurations and run the quantum models for 100 epochs as budget. As a performance measure, we recorded the best validation accuracy obtained over 100 epochs. Fig. 7.3 shows the distribution of the 10-fold cross-validation accuracy obtained per dataset. We observe the impact of hyperparameter optimization by the difference between the least performing and the best model configuration. For instance, on the wilt dataset, the best model gets an accuracy close to 1, and the least below 0.25. We can also see that some datasets present a smaller spread of performances. ilpd and blood-transfusion-service-center are in this case. It seems that hyperparameter optimization does not have a real effect, because most hyperparameter configurations give the same result. As such, the surrogates could not differentiate between various configurations. In general, hyperparameter optimization is important for getting high performances per dataset and detecting datasets where the importance study can be applied.

### 7.5.2 Surrogate verification

Functional ANOVA relies on an internal surrogate model to determine the marginal contribution per hyperparameter. If this surrogate model is not accurate, this can have a severe limitation on the conclusions drawn from functional ANOVA. In this experiment, we verify whether the hyperparameters can explain the performances of the models. Table 7.3 shows the performance of the internal surrogate models. We notice low regression scores for the two datasets (less than 0.75 R2 scores). Hence we remove them from the analysis.

### 7.5.3 Marginal contributions

For functional ANOVA, we used 128 trees for the surrogate model. Fig. 7.4(a,b) shows the marginal contribution of each hyperparameter over the remaining 5 datasets. We distinguish 3 main levels of importance. According to these results, the learning rate, depth, and the data encoding circuit and reuploading strategy are critical. These results are in line with our expectations. The entangler gate, connectivity, and whether we use $R_X$ gates in the variational layer are the least important according to functional ANOVA. Hence, our results reveal new insights into these hyperparameters that are not considered in general.




**(a)** **(b)**

Figure 7.4: The marginal contributions per dataset are presented as a) the variance contribution and b) the difference between the minimal and maximal value of the marginal of each hyperparameter. The hyperparameters are sorted from the least to most important using the median. We distinguish from the plot 3 main levels of importance.

### 7.5.4 Random search verification

In line with the work of [192], we perform an additional verification experiment that verifies whether the outcomes of functional ANOVA are in line with our expectations. However, the verification procedure involves an expensive, post-hoc analysis: a random search procedure fixing one hyperparameter at a time. As our quantum simulations are costly, we used the surrogate models fitted on the current dataset considered over the

1 000 configurations obtained initially to predict the performances one would obtain when presented with a new configuration.

Fig. 7.5 shows the average rank of each run of random search, labeled with the hyperparameter whose value was fixed to a default value. A high rank implies poor performance compared to the other configurations, meaning that tuning this hyperparameter would have been important. We witness again the 3 levels of importance, with almost the same order obtained. However, the input_activation_function is deemed more important while the batch size is less.

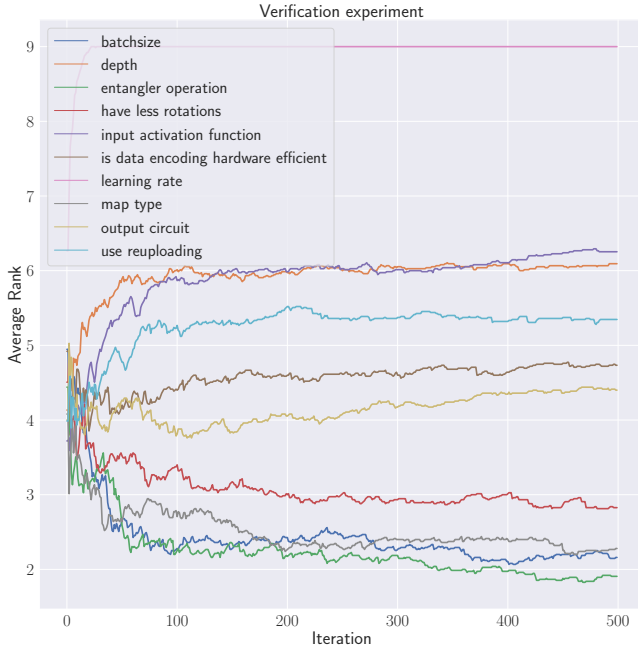



Figure 7.5: Verification experiment of the importance of the hyperparameters. A random search procedure up to 500 iterations excluding one parameter at a time is used. A lower curve means the hyperparameter is deemed less important.

More simulations with more datasets may be required to validate the importance. However, we retrieve empirically the importance of well-known hyperparameters while considering less important ones. Hence functional ANOVA becomes an interesting tool for quantum machine learning in practice.

## 7.6 Conclusion

In this chapter, we study the importance of hyperparameters related to quantum neural networks for classification using the functional ANOVA framework. Our experiments are carried out over OpenML datasets that match the current scale of quantum hardware simulations (i.e., datasets that have at most 20 features after pre-processing operators have been applied, hence using 20 qubits). We selected and presented the hyperparameters from an aggregation of quantum computing literature and software. Firstly, hyperparameter optimization highlighted datasets where we observed high differences between configurations. This underlines the importance of hyperparameter optimization for these datasets. There were also datasets that showed little difference. These led us to extend the methodology by adding an additional verification step of the internal surrogate performances. From our results, we distinguished 3 main levels of importance. On the one hand, Adam's learning rate, depth, and data encoding strategy are deemed very important, as we expected. On the other hand, the less considered hyperparameters such as the particular choice of the entangling gate and using 3 rotation types in the variational layer are in the least important group. Hence, our experiment both confirmed expected patterns and revealed new insights for quantum model selection.

For future work, further methods from the field of automated machine learning can be applied to quantum neural networks [31, 60, 133]. Indeed, our experiments have shown the importance of hyperparameter optimization, and this should become part of the protocols applied within the community. We further envision functional ANOVA to be employed in future works related to quantum machine learning and understanding how to apply quantum models in practice. For instance, it would be interesting to consider quantum data, for which quantum machine learning models may have an advantage. Plus, extending hyperparameter importance to techniques for scaling to a large number of features with the number of qubits, such as dimensionality reduction or divide-and-conquer techniques, can be left for future work. Finally, this type of study can also be extended to different noisy hardware and to algorithm/model selection and design. If we have access to a cluster of different quantum computers, then choosing which hardware works best for machine learning tasks becomes possible. One could also extend our work with meta-learning [31], where a model configuration is selected based on meta-features created from dataset features. Such types of studies already exist for parameterized quantum circuits applied to combinatorial optimization and we presented them in chapters 3 and 4 [135, 138].

Table 7.1: List of hyperparameters considered for hyperparameter importance for quantum neural network, as we named them in our Tensorflow-Quantum code.

| Hyperparameter | Values | Description |
|---|---|---|
| Adam learning rate | $[10^{-4}, 0.5]$ (log) | The learning rate with which the quantum neural network starts training. The range was taken from the automated machine learning library Auto-sklearn [60]. We uniformly sample taking the logarithmic scale. |
| batch size | 16, 32, 64 | Number of samples in one batch of Adam used during training |
| depth | $\{1, 2, \cdots, 10\}$ | Number of variational layers defining the circuit |
| is data_encoding hardware efficient | True, False | Whether we use the hardware-efficient circuit $\bigotimes_{i=1}^{n} R_X(x_i)$ or an IQP circuit defined in Eq.7.1 to encode the input data. |
| use reuploading | True, False | Whether the data encoding layer is used before each variational layer or not. |
| have less rotations | True, False | If True, only use layers of $R_Y, R_Z$ gates as the variational layer. If False, add a layer of $R_X$ gates. |
| entangler operation | cz, sqiswap | Which entangling gate to use in $U_{\text{ent}}$ |
| map type | ring, full, pairs | The connectivity used for $U_{\text{ent}}$. The ring connectivity use an entangling gate between consecutive indices $(i, i+1), i \in \{1, \ldots, n\}$ of qubits. The full one uses a gate between each pair of indices $(i, j), i < j$. Pairs connect even consecutive indices first, then odd consecutive ones. |
| input activation function | linear, tanh | Whether to input $tanh(x_i)$ as rotations or just $x_i$. |
| output circuit | 2Z, mZ | The observable(s) used as output(s) of the circuit. If 2Z, we use all possible pairs of qubit indices defining $Z \otimes Z$. If mZ, the tensor product acts on all qubits. Note we do not use single-qubit $Z$ observables although they are quite often used in the literature. Indeed, they are provably not using the entire circuit when it is shallow. Hence we decided to use $Z \otimes Z$ instead. Also, a single neuron layer with a sigmoid activation function is used as a final decision layer similar to [168]. |

Table 7.2: List of datasets used in this study. The number of features is obtained after a usual preprocessing used in machine learning methods, such as one-hot encoding.

| Dataset | OpenML Task ID | Number of features | Number of instances |
|---|---|---|---|
| breast-w | 15 | 9 | 699 |
| diabetes | 37 | 8 | 768 |
| phoneme | 9952 | 5 | 5 404 |
| ilpd | 9971 | 11 | 583 |
| banknote-authentication | 10093 | 4 | 1 372 |
| blood-transfusion-service-center | 10 101 | 4 | 748 |
| wilt | 146820 | 5 | 4 839 |

Table 7.3: Performances of the surrogate models built within functional ANOVA over a 10-fold cross-validation procedure. We present the average coefficient of determination (R2), root mean squared error (RMSE), and Spearman's rank correlation coefficient (CC). These are common regression metrics for benchmarking surrogate models on hyperparameters [53]. The surrogates over ilpd and blood-transfusion-service-center obtain low scores (less than .75 R2), hence we remove them from the study.

| Dataset | R2 score | RMSE | CC |
|---|---|---|---|
| breast-w | 0.8663 | 0.0436 | 0.9299 |
| diabetes | 0.7839 | 0.0155 | 0.8456 |
| phoneme | 0.8649 | 0.0285 | 0.9282 |
| ilpd | 0.1939 | 0.0040 | 0.4530 |
| banknote-authentication | 0.8579 | 0.0507 | 0.9399 |
| blood-transfusion-service-center | 0.6104 | 0.0056 | 0.8088 |
| wilt | 0.7912 | 0.0515 | 0.8015 |