

Resource Saving via Ensemble Techniques for Quantum Neural Networks

Massimiliano Incudini^{1,*}, Michele Grossi², Andrea Ceschini³,
Antonio Mandarino⁴, Massimo Panella³, Sofia Vallecorsa², and
David Windridge⁵

¹Dipartimento di Informatica, Università di Verona, Strada Le
Grazie, 15, Verona, 37134, Italy

²European Organization for Nuclear Research (CERN), Espl. des
Particules, 1, Meyrin, 1211, Switzerland

³Dipartimento di Ingegneria dell'Informazione, Elettronica e
Telecomunicazioni (DIET), Università di Roma "La Sapienza", Via
Eudossiana, 18, Roma, 00184, Italy

⁴International Centre for Theory of Quantum Technologies
(ICTQT), University of Gdansk, Jana Bażyńskiego 1A, Gdańsk,
80-309, Poland

⁵Department of Computer Science, Middlesex University, The
Burroughs, London, NW4 4BT, United Kingdom

*Correspondence to: massimiliano.incudini@univr.it

Abstract

Quantum neural networks hold significant promise for numerous applications, particularly as they can be executed on the current generation of quantum hardware. However, due to limited qubits or hardware noise, conducting large-scale experiments often requires significant resources. Moreover, the output of the model is susceptible to corruption by quantum hardware noise. To address this issue, we propose the use of ensemble techniques, which involve constructing a single machine learning model based on multiple instances of quantum neural networks. In particular, we implement bagging and AdaBoost techniques, with different data loading configurations, and evaluate their performance on both synthetic and real-world classification and regression tasks. To assess the potential performance improvement under different environments, we conduct experiments on both simulated, noiseless software and IBM superconducting-based QPUs, suggesting these techniques can mitigate the quantum hardware noise. Additionally, we quantify the amount of resources saved using these ensemble techniques. Our findings indicate that these methods enable the

construction of large, powerful models even on relatively small quantum devices.

1 Introduction

The emerging field of quantum machine learning [1] holds promise for enhancing the accuracy and speed of machine learning algorithms by utilizing quantum computing techniques. Although the potential of quantum machine learning is expected to be advantageous for certain classes of problems in chemistry, physics, material science, and pharmacology [2], its applicability to more conventional use cases remains uncertain [3]. Notably, utilizable quantum machine learning algorithms generally need to be adapted to run on ‘NISQ’ devices [4], that are current noisy quantum computer, no error corrected and with modest number of qubits and circuit depth capabilities. In the quantum machine learning scenario, the quantum counterparts of classical neural networks, quantum neural networks [5], have emerged as the de facto standard model for solving supervised and unsupervised learning tasks in the quantum domain.

While quantum neural networks have generated much interest, they presently have some issues. The first is *barren plateau* [6] characterised by the exponentially-fast decay of the loss gradient’s variance with increasing system size. This problem may be exacerbated by various factors, such as having overly-expressive quantum circuits [7]. To address this issue, quantum neural networks need to be carefully designed [8] and to incorporate expressibility control techniques such as projection [9] and bandwidth control [10]. The second problem, which is the one addressed in this work, concerns the amount of resources required to run quantum neural networks (the limited number of total qubits -currently up to over a hundred- and the low fidelity of operations on current quantum devices severely restrict the size of the quantum neural network in terms of input dimension and layers).

In order to address the latter issue, we propose employing of NISQ-appropriate implementation of ensemble learning [11], a widely used technique in classical machine learning for tuning the bias and variance of a specific machine learning mechanism via the construction of a stronger classifier using multiple weak components, such that the ensemble, as a whole, outperforms the best individual classifier. The effectiveness of ensemble systems has been extensively demonstrated empirically and theoretically [12], although there does not currently exist any overarching theoretical framework capable of e.g. covering the requirements of ensemble components diversity to guarantee its out-performance. We here seek to provide and quantify a motivation for employing classical ensemble techniques in relation to NISQ-bases quantum neural networks, which we address via the following three arguments.

The first argument concerns the potential for the superior performance of an ensemble system composed of small quantum neural networks compared to a single larger quantum neural network. This notion is based on the rationale that while quantum neural networks are inherently powerful machine learning

models, they exhibit intrinsic variance due to the nature of highly non-convex loss landscape, implying that different predictors will result from randomly-initialised stochastic gradient descent training, in common with classical neural networks. (Modern deep learning practice often deliberately overparameterises the network in order to render the loss more convex [13], with the asymptotic case of infinitely wide neural networks exhibiting a fully convex loss landscape, making it effectively a linear model [14]). Although overparameterization in quantum neural networks has been studied theoretically [15, 16, 17] and has been shown to be beneficial to generalization performances within certain settings, the increase in resource requirements makes this approach almost completely impractical on NISQ devices. In the classical literature, however, it has been demonstrated that ensemble techniques can perform comparably to the largest (generally overparameterized) models with significantly fewer resources (especially in relation to overall model parameterization), c.f. for example [18, Figure 2].

The second argument pertains to the resource savings achievable by ensemble systems, particularly in terms of the number of qubits, gates, and training samples required. For example, the boosting ensemble technique involves progressive dividing of the training dataset into multiple, partially overlapping subsets on the basis of their respective impact on the performance of the cumulative ensemble classifier created by summing of the partial weak classifiers trained on previously-selected data subsets. This enables the ensemble quantum neural network to be constructed in parallel with individual quantum neural networks operating on datasets of reduced size. The random subspace technique, by contrast, trains each base predictor on a random subset of features, but also provides an advantage in terms of the overall number of qubits and gates required. Employing the random subspace technique in a quantum machine learning setting would parallel the various quantum circuit splitting techniques (c.f. for example [19]), and divide-and-conquer approaches, that have been utilized in the field of quantum chemistry [20] and quantum optimization [21].

Our third argument, which is specific to quantum computing, examines the potential of ensembles’ noise-canceling ability. Previous works have demonstrated that ensembles can enhance the performance of several noisy machine-learning tasks (see [22]). Our investigation aims to determine whether and to what extent these techniques can reduce the impact of noise during the execution on a NISQ device *at the applicative level*. This approach differs from most current approaches, which aim to reduce noise at a lower level, as described in [23].

We here examine the impact of ensemble techniques based on bagging (bootstrap aggregation) and boosting ensembles in a quantum neural network setting across seven variant data loading schemes. Bagging techniques are selected for their applicability in high-variance settings, i.e. those exhibiting significant fluctuations in relation to differ initialisations and differ sample subselections; contrarily, boosting techniques are effective in relation to high-bias models, i.e. those which are relatively insensitive to data subsampling.

Our first objective is to quantify the amount of resources (in particular, the number of qubits, gates, parameters, and training samples) saved by the respective approaches. Secondly, we evaluate the performance using quantum

neural networks as base predictors to solve a number of representative synthetic and real-world regression and classification tasks. Critically, the accuracy and loss performance of these approaches are assessed with respect to the number of layers of the quantum neural networks in a simulated environment. We thus obtain a layer-wise quantification of performance that addresses one of the fundamental questions in architecting deep neural systems, namely, how many layers of abstraction to incorporate? Note that this question is fundamentally different in a quantum setting compared to classical neural systems; in the latter, the possibility of multi-level feature learning exists, and thus the potential for indefinite performance improvement with neural layer depth [17]. This contrast with the quantum neural networks, in which an increase in the number of layers affects the expressibility of the ansatz and thus might introduce a barren plateau [7].

Finally, the noise-canceling capabilities of ensembles will be investigated by testing a synthetic linear regression task on IBM’s superconductor-based quantum processing unit (QPU) Lagos.

Contributions Our contributions are the following:

- We evaluate various ensemble schemes that incorporate bagging and boosting techniques into quantum neural networks, and quantify the benefits in terms of resource savings, including the number of qubits, gates, and training samples required for these approaches.
- We apply our approach to the IBM Lagos superconductor-based quantum processing unit to investigate the potential advantages of bagging techniques in mitigating the effects of noise during the execution of quantum circuits on NISQ devices.
- We conduct a layer-wise analysis of quantum neural network performance in the ensemble setting with a view to determining the implicit trade-off between ensemble advantage and layer-wise depth.

2 Related Works

The quest for quantum algorithms able to be executed on noisy small-scale quantum systems led to the concept of Variational Quantum Circuits (VQCs), i.e. quantum circuits based on a hybrid quantum-classical optimization framework [24, 25]. VQCs are currently believed to be promising candidates to harness the potential of QC and achieve a quantum advantage [26, 27, 28]. VQCs rely on a hybrid quantum-classical scheme, where a parameterized quantum circuit is iteratively optimized with the help of a classical co-processor. This way, low-depth quantum circuits can be efficiently designed and implemented on the available NISQ devices; the noisy components of the quantum process are mitigated by the low number of quantum gates present in the VQCs. The basic structure of a VQC include a data encoding stage, where classical data are embedded

into a complex Hilbert space as quantum states, a processing of such quantum states via an ansatz made of parameterized rotation gates and entangling gates, and finally a measurement of the circuit to retrieve the expected outcome. Many different circuit architectures and ansatzes have been proposed for VQCs [29, 30, 31, 32], depending on the structure of the problem or on the underlying quantum hardware. VQCs demonstrated remarkable performances and a good resilience to noise in several optimization tasks and real-world applications. For example, researchers in [33] introduced a circuit-centric quantum classifier based on VQC that could effectively be implemented on a near-term quantum device. It correctly classified quantum encoded data and demonstrated to be robust against noise. Authors in [25] proposed a VQC that successfully approximated high-dimensional regression and classification functions with a limited number of qubits.

VQCs are incredibly well-suited for the realization of quantum neural networks with a constraint on the number of qubits [34]. A quantum neural network is usually composed of a layered architecture able to encode input data into quantum states and perform heavy manipulations in a high-dimensional feature space. The encoding strategy and the choice of the circuit ansatz are critical for the achievement of superior performances over classical NNs: more complex data encoding with hard-to-simulate feature maps could lead to a concrete quantum advantage [35], but too expressive quantum circuits may exhibit flatter cost landscapes and result in untrainable models [7]. An example of quantum neural network was given in [36], where a shallow NN was employed to perform classification and regression tasks using both simulators and real quantum devices. In [37], authors proposed a multi-layer Quantum Deep Neural Network (QDNN) with three variational layers for an image classification task. They managed to prove that QDNNs have more representation capacity with respect to classical deep NN. A hybrid Quantum-classical Recurrent Neural Network (QRNN) was presented in [38] to solve a time series prediction problem. The QRNN, composed of a quantum layer as well as two classical recurrent layers, demonstrated superior performances over the classical counterpart in terms of prediction error.

However, quantum neural networks suffer from some non-negligible problems, which deeply affect their performances and limit their impact in the quantum ecosystem. Firstly, they are still subject to quantum noise, and it gets worse as the number of layers (i.e., the depth of the quantum circuit) increases [39, 40]. Secondly, barren plateaus phenomena may occur depending on the ansatz and the number of qubits chosen, reducing the trainability of such models [7, 41, 6]. Finally, data encoding on NISQ devices continues to represent an obstacle when the number of features is considerable [34], making them hard to implement and train [38].

In classical ML, ensemble learning has been investigated for years to improve generalization and robustness over a single estimator [42, 11]. Ensembling is based on the so-called “wisdom of the crowd” principle, namely it combines the predictions of several base estimators with the same learning algorithm to build a single stronger model. Despite there are many different ensemble methods, the latter can be easily grouped into two different categories: bagging methods, which

build and train several estimators independently and then compute an average of their predictions [43], and boosting methods, which in turn train the estimators sequentially so that each one corrects the predictions of the prior models and output a weighted average of such predictions [44]. Ensemble methods for NNs have also been extensively studied, yielding remarkable performances in both classification and regression tasks [45, 46, 47].

In the quantum setting, the adoption of an ensemble strategy has received little consideration in the past few years, with very few approaches focusing on near-term quantum devices and VQC ensembles. In [48, 49], the authors exploit the superposition principle to obtain an exponentially large ensemble wherein each instance is weighted according to its accuracy on the training dataset. However, they make use of a fault-tolerant approach rather than considering limited quantum resources. A similar approach is explored in [50], where authors create an ensemble of Quantum Binary Neural Networks (QBNNs) with reduced computational training cost without taking into consideration the amount of quantum resources necessary to build the circuit. An efficient strategy for bagging with quantum circuits is proposed in [51] instead. Very recently, [52] has proposed a distributed framework for ensemble learning on a variety of NISQ quantum devices, although it requires many NISQ devices to be actually implemented. A quantum ECOC multiclass ensemble approach was proposed in [53]. In [54], the authors investigated the performance enhancement of a majority-voting-based ensemble system in the quantum regime. Authors in [55] studied the role of ensemble techniques in the context of quantum reservoir computing. Finally, an analysis of robustness to hardware error as applied to quantum reinforcement learning, and presenting compatible results, is given in [56].

In this paper, we propose a classical ensemble learning approach to the outputs of several quantum neural networks in order to reduce the quantum resources for a given quantum model and provide superior performances in terms of error rate over single quantum neural network instances. To the best of our knowledge, no one has ever proposed such an ensemble framework for VQCs. We also compare both bagging and boosting strategy to provide an analysis on the most appropriate ensemble methods for quantum neural networks in a noiseless setting. An error analysis with respect to the number of layers of the quantum neural networks reveals that bagging models greatly outperform the baseline model with low number of layers, with remarkable performances as the number of layers increase. Finally, we apply our approach to the IBM Lagos superconductor-based QPU to investigate the potential advantages of bagging techniques in mitigating the effects of noise during the execution of quantum circuits on NISQ devices.

3 Background and Notation

We provide a brief introduction to the notation and concepts used in this work. The sets \mathcal{X} and \mathcal{Y} represent the set of features and targets, respectively. Typically,

\mathcal{X} is equal to \mathbb{R}^d , with d equal to the dimensionality in input, whereas \mathcal{Y} is equal to \mathbb{R} for regression tasks and \mathcal{Y} is equal to $\{c_1, \dots, c_k\}$ for k -ary classification tasks. Sequences of elements are indexed in the apex with $x^{(j)}$, where the i -th component is denoted as x_i . The notation $\epsilon \sim \mathcal{N}(\mu, \sigma^2)$ indicates that the value of ϵ is randomly sampled from a univariate normal distribution with mean μ and variance σ^2 . We use the function $\llbracket P \rrbracket$ to denote one when the predicate P is true and zero otherwise.

3.1 Models in quantum machine learning

We define the state of a quantum system as the density matrix ρ having unitary trace and belonging to the Hilbert space $\mathcal{H} \equiv \mathbb{C}^{2^n \times 2^n}$ where n is the number of qubits. The system starts in the state $\rho_0 = |0\rangle\langle 0|$. The evolution in a closed quantum system is described by a unitary transformation $U = \exp(-itH)$, $t \in \mathbb{R}$, H Hermitian operator, and acts like $\rho \mapsto U^\dagger \rho U$. The measurement of the system in its computational basis $\{\Pi_i = |i\rangle\langle i|\}_{i=0}^{2^n-1}$ applied to the system in the state ρ will give outcome $i \in 0, 1, \dots, 2^n - 1$ with probability $\text{Tr}[\Pi_i \rho \Pi_i]$ after which the state collapses to $\rho' = \Pi_i \rho \Pi_i / \text{Tr}[\Pi_i \rho \Pi_i]$. A different measurement operation is given by the expectation value of an observable $O = \sum_i \lambda_i \Pi_i$ acting on the system in state ρ , whose value is $\langle O \rangle = \text{Tr}[\rho O]$.

Quantum computation can be described using a quantum circuit, a sequence of gates (i.e. elementary operations) acting on one or more qubits of the system terminating with the measurement operation over some or all of its qubits. The output of the measurement can be post-processed using a classical function. "The set of gates available shall be *universal*", i.e. the composition of such elementary operation allows the expression of any unitary transformation with arbitrary precision. An exemplar universal gate set is composed of parametric operators $R_x^{(i)}(\theta) = \exp(-i\frac{\theta}{2}\sigma_x^{(i)})$, $R_y^{(i)}(\theta) = \exp(-i\frac{\theta}{2}\sigma_y^{(i)})$, $R_z^{(i)}(\theta) = \exp(-i\frac{\theta}{2}\sigma_z^{(i)})$, and the operator CNOT $^{(i,j)} = \exp(-i\frac{\pi}{4}(I - \sigma_z^{(i)})(I - \sigma_x^{(j)}))$. The gate I is the identity. The matrices $\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $\sigma_y = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$, $\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ are the Pauli matrices. The apex denotes explicitly the qubits in which the transformation acts.

Quantum machine learning forms a broad family of algorithms, some of which require fault-tolerant quantum computation while others are ready to execute on current generation 'NISQ' (noisy) quantum devices. The family of NISQ-ready techniques of interest in this document is denoted *variational quantum algorithms* [24]. These algorithms are based on the tuning of a cost function $C(\theta)$ dependent on a set of parameters $\theta \in [0, 2\pi]^P$ and optimized classically (possibly via gradient descent-based techniques) to obtain the value $\theta^* = \arg \min_{\theta} C(\theta)$. Optimization through gradient-descent thus involves computation of the gradient of C . This can be done using finite difference methods or else the parameter-shift rule [57]. The parameter-shift rule is particularly well-suited for NISQ devices as it can utilise a large step size relative to finite difference methods, making it less sensitive to noise in calculations.

In general, $C(\theta)$ is a function corresponding to a parametric quantum transformation $U(\theta)$ of a length polynomial in the number of qubits, the set of input

states $\{\rho_i\}$, and the set of observables $\{O_k\}$. Specifically, a *quantum neural network* is a function in the form

$$f(x; \theta) = \text{Tr}[U^\dagger(\theta)V^\dagger(x)\rho_0V(x)U(\theta)O] \quad (1)$$

where ρ_0 is the initial state of the system, $V(x)$ is a parametric quantum circuit depending on the input parameters $x \in \mathcal{X}$, $U(\theta)$ is a parametric quantum circuit named an *ansatz* that depends on the trainable parameters $\theta \in [0, 2\pi)^P$, and O is an observable. Given the training dataset $\{(x^{(i)}, y^{(i)})\}_{i=1}^M \in (\mathcal{X} \times \mathcal{Y})^M$, the cost function of a quantum neural network, being a supervised learning problem, is the empirical risk

$$C(\theta) = \sum_{i=1}^M \ell(f(x^{(i)}; \theta), y^{(i)}) \quad (2)$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ is any convex loss function, e.g. the mean square error.

The quantum neural network constitutes a linear model in the Hilbert space of the quantum system as a consequence of the linearity of quantum dynamics. It behaves, in particular, as a *kernel machine* that employs the unitary $V(x)$ as the feature map $\rho \mapsto \rho_x = V(x)\rho$, while the variational ansatz $\rho \mapsto \rho_\theta = U(\theta)\rho$ adjusts the model weights. Note that although the model is linear in the Hilbert space of the quantum system, the measurement projection makes it nonlinear in the parameter space, enabling a set of rich dynamics. quantum neural networks can have a layer-wise structure, i.e., $U(\theta) = \prod_{i=1}^{\ell} U_i(\theta_i)$, which provides it with further degrees of freedom for optimization (however, due to the lack of nonlinearity between the layers, the model does not possess the hierarchical feature learning capabilities of classical neural networks).

The selection of the ansatz is thus a crucial aspect in defining the quantum neural network, and it is required to adhere to certain classifier-friendly principles. Expressibility is one such, being the property governing the extent of the search space that can be explored by the optimization method. Although there are various ways to formalize expressibility, one of the most widely used definitions is based on the generation of state ensembles $\{\rho_\theta = U(\theta)\rho_0 \mid \theta \in \Theta\}$ that are similar to Haar-random (i.e. uniform) distributions of states. Expressible unitaries are those for which the operator norm of a certain expression involving the Haar measure and the state ensemble is small. However, expressible circuits are susceptible to the barren plateau problem, where the variance of the gradient decreases exponentially with the number of qubits, making parameter training infeasible. The varieties of ansatz and their expressibilities are presented in [58]. Expressibility is tightly connected to the concept of controllability in quantum optimal control, and authors in [8] show that the asymptotic limit of the number of layers $\ell \rightarrow \infty$ in the expressible circuits are the controllable ones, i.e. those whose ansatz is underlied by a Lie algebra matching the space of skew-Hermitian matrices $\mathfrak{u}(2^n)$.

Data selection strategy			
		Subset of features	
		No	Yes
Subset of samples	No	/	Random subspace
	Yes	Bootstrapping, pasting	Random patch

Composition + training of single model instances			
		Model instances	
		Heterogeneous	Homogeneous
Type of processing	Sequential	/	Boosting
	Parallel	Stacking	Bagging

Combination rule of the outputs			
		Discrete	Continuous
		Majority voting	Average
		Weighted majority voting	Weighted average
		Borda counts	Min Max

Figure 1: Taxonomy of the three aspects characterizing an ensemble system.

3.2 Ensemble techniques

The purpose of using ensemble systems is to improve the generalization performance through reducing the bias or variance of a decision system. Such a result is obtained by training several models and combining the outcomes according to a combination rule. A large body of literature on ensemble techniques exists; the reader is referred to [11] for a general overview.

The idea behind the ensemble system may be motivated by Condorcet’s jury theorem [12]: a jury of m peers, each having probability $p = \frac{1}{2} + \epsilon$, $0 < \epsilon \ll 1$, of giving the correct answer, implies that the probability of the verdict given by majority voting to be correct is

$$p_{\text{jury}} = \sum_{k=\lceil m/2 \rceil+1}^m \binom{m}{k} p^k (1-p)^{m-k} \quad (3)$$

and quickly approaches 1 as $m \rightarrow \infty$. The theorem, broadly interpreted, suggests that a combination of small, individually ineffective machine learning models h_1, \dots, h_m (*weak learners*) can be combined to constitute a more powerful one, with arbitrarily good performance depending on the nature of data manifold and the base classifiers h_{ens} (*strong learner*). According to [11], three aspects characterize an ensemble system: a data selection strategy, the composition plus training strategies of the single model instances, and the combination rule of its output. Some of the possible choices are summarized in Figure 1.

The data selection strategy determines how the data should be distributed to the individual instances. If all instances are trained on the same dataset, their predictions will be highly correlated, resulting in similar output. The *bootstrapping* technique creates smaller, overlapping subsets by sampling with

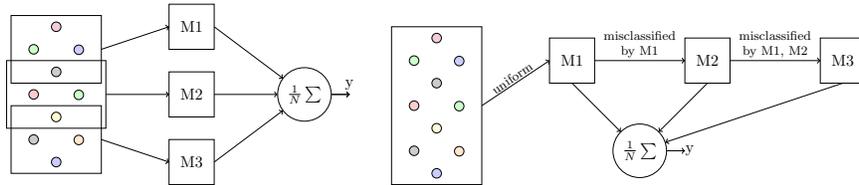


Figure 2: Comparison between bagging (left) and ‘vanilla’ boosting (right) techniques. The bagging ensemble trains the models in parallel over a subset of the dataset drawn uniformly; each prediction is then merged via an average function. The boosting ensemble trains the models sequentially, the first predictor draws the samples uniformly, and the subsequent models draw the elements from a probability distribution biased toward previously misclassified items.

replacement from the dataset, which are then assigned to different instances. Alternatively, the *pasting* technique can be used for processing larger datasets by subsampling without replacement. Another approach is to divide the dataset by randomly assigning different sets of features with replacement, known as the random subspace technique (when the bootstrapping and random subspace techniques are combined, the result is the *random patch* technique).

There are numerous schemes for combining predictors, with *bagging* being the most straightforward and commonly used. Bagging, short for bootstrap aggregation, involves the creation of multiple homogeneous model instances trained on bootstrapped datasets. An instance of a bagging scheme is the random forest, which involves bagging decision trees trained on differing sample subsets (in some cases, random forests may favor a random patch data selection strategy over bagging). Another predictor combination scheme is *boosting*, which involves training a sequence of predictors via subsampling data according to the following strategy: an initial predictor is trained on a uniformly drawn subset of samples, while the i -th instance of the predictor is trained on a subset of elements that the previous ensemble classifier incorrectly predicted. The ensemble is itself the convex cumulative sum over predictors. Numerous variations of boosting exist, one of the most notable being AdaBoost [59]. Contrary to vanilla boosting, AdaBoost employs an exponential loss such that the ensemble error function allows for the fact that it is only the sign of outcome that is significant. These two scheme are illustrated in Figure 2. The other major ensemble scheme is *stacking* in which a collection of heterogeneous classifiers trained on the same dataset are combined via an optimised meta-classifier.

The combination rule merges the output of individual models h_1, \dots, h_m . In classification tasks i.e. where the label output is discrete $y \in C = \{c_1, \dots, c_k\}$, the most commonly used rule is majority voting. This is calculated as $y_{\text{ens}} = \arg \max_{c \in C} \sum_{i=1}^m \mathbb{1}[h_i(x) = c]$. Where there exists prior knowledge regarding the performance of individual predictors, positive weights w_i can be assigned, such that the output is a weighted majority vote. The ensemble prediction in this case will be $y_{\text{ens}} = \arg \max_{c \in C} \sum_{i=1}^m w_i \mathbb{1}[h_i(x) = c]$. Alternatively, the

borda count method sorts labels in descending order by likelihood, with the ensemble prediction being the highest ranking sum. Nevertheless, averaging functions can also be utilised for ensemble classifiers. For regression tasks where $y \in \mathbb{R}$, common combination rules are (possibly weighted) mean, minimum, and maximum.

4 Discussion

Ensemble techniques, while well-established in the classical realm, have been largely overlooked in the quantum literature, leaving a number of open questions in this setting, such as whether bagging techniques, which reduce variance, can be deployed as effectively as boosting techniques, which reduce bias (both of which are also data-manifold and base-model dependent). It is also unclear as to the relative resource saving in terms of circuit size (number of qubits) and depth (number of gates), and also samples required for training, that can be obtained by using an ensemble of quantum neural networks instead of a single, large quantum network. Furthermore, it is not currently well understood the extent to which an ensemble system can mitigate hardware noise. Our experiments are designed to explore these questions.

To investigate the first two aspects, we conduct a suite of experiments within a simulation environment, employing seven distinct ensemble schemes with varying strategies for data selection, model training and decision combination applied to four synthetic and real-world datasets, encompassing both regression and classification tasks. Specifically, we analyze: a synthetic linear regression dataset, the Concrete Compressive Strength regression dataset, the Diabetes regression dataset, and the Wine classification dataset, which are widely used benchmarks for evaluating machine learning models.

Six of the proposed techniques are classified as bagging methods, employing bootstrapped data to generate the ensemble, while the seventh is a sequential boosting technique, namely AdaBoost. In particular, we implemented the AdaBoost.R2 version [60] for the regression tasks and the AdaBoost SAMME.R version [61] for the classification problem. The bagging ensembles are characterized by two parameters: the sample ratio $r_n \in [0, 1]$, which determines the percentage of training samples used for each base predictor (with replacement), and the feature ratio $r_f \in [0, 1]$, which indicates the percentage of features used for each predictor (without replacement). We test six bagging schemes by varying $(r_n, r_f) \in \{0.2, 1.0\} \times \{0.3, 0.5, 0.8\}$. For both the classification and regression tasks, the outputs of the base predictors are combined via averaging. In the case of the AdaBoost ensemble, the training set for each base predictor has the same size and dimensionality as the original training set. However, the samples are not uniformly drawn but are selected and weighted based on the probability of misclassification by previous classifiers composing the cumulative ensemble; single predictors are hence combined using a weighted average. Each ensemble system comprises 10 base predictors. The characteristics of these ensemble schemes are summarized in Table 1, where FM identifies the baseline quantum neural network

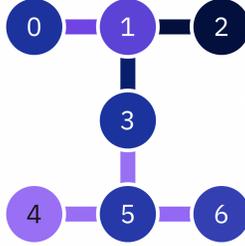


Figure 3: Topology of IBM Lagos quantum processing unit

model, whereas $\text{Bag}_{r_f-r_n}$ represents a bagging model with r_f percentage of the features and r_n percentage of the samples. Our experiments aim to evaluate the performance of each of the ensemble frameworks in comparison to the baseline model, as well as to assess the overall resource saving, including the number of qubits and overall parametric requirements.

Model	Data Loading		Ensemble	#BP	Rule
	RSBS (r_f)	BST (r_n)			
FM	-	-	-	-	-
Bag_0.3_0.2	0.3	0.2	Bagging	10	Avg
Bag_0.3_1.0	0.3	1.0	Bagging	10	Avg
Bag_0.5_0.2	0.5	0.2	Bagging	10	Avg
Bag_0.5_1.0	0.5	1.0	Bagging	10	Avg
Bag_0.8_0.2	0.8	0.2	Bagging	10	Avg
Bag_0.8_1.0	0.8	1.0	Bagging	10	Avg
AdaBoost	1.0	1.0	AdaBoost	10	W.Avg

Table 1: Characteristics of the baseline benchmark model (0) and ensemble systems (I to VII). The ensemble system is identified by its broad data loading method (BST for Boosting and RSBS for Random Subspace), predictor composition & training type (Ensemble), number of base predictors (#BP), composition rule (Rule, with Avg representing the average function and W.Avg representing weighted average).

To investigate the impact of quantum hardware noise, we conduct additional experiments on the IBM Lagos QPU. Such a device is a 7-qubit superconducting-based quantum computer. The topology of Lagos is depicted in Figure 3. Specifically, we compare the performance of the baseline model FM with that of the Bag_0.8_0.2 configuration on the linear regression dataset. Our goal is to determine whether ensemble techniques can effectively mitigate quantum noise, and whether the difference in performance between single predictors and ensemble systems is more pronounced within a simulated environment in comparison with real-world execution on quantum hardware.

4.1 Experimental setup

This section outlines experimental protocols used to evaluate the performance of the various ensemble approaches in terms of both the experimental structure and specific parameters/settings used to configure the algorithm and hardware.

Choice of quantum neural networks We utilize a quantum neural network of the form $f(x; \theta) = \text{Tr}[U^\dagger(\theta)V^\dagger(x)\rho_0V(x)U(\theta)O]$, which operates on n qubits, with n corresponding to the number of features in the classification/regression problem. For the feature map, we opted for the simple parametric transformation $V(x) = \bigotimes_{i=1}^n R_y^{(i)}(x_i)$. This choice was motivated by the findings in [62], suggesting that more complex feature maps can lead to unfavorable generalization properties, incorporation of which may thus unnecessarily bias our findings. (In [63], various feature maps are compared).

The ansatz is implemented with the parametric transformations structured layer-wise with, for ℓ the number of layers, a total of $3\ell n$ parameters. It is thus defined as:

$$U_\ell(\theta) = \prod_{k=1}^{\ell} \left[\left(\bigotimes_{i=1}^n R_x^{(i)}(\theta_{3kn+2n+i}) \right) \left(\prod_{i=1}^{n-1} \text{CX}^{(i,i+1)} \right) \left(\bigotimes_{i=1}^n R_z^{(i)}(\theta_{3kn+n+i}) \right) \right. \\ \left. \left(\prod_{i=1}^{n-1} \text{CX}^{(i,i+1)} \right) \left(\bigotimes_{i=1}^n R_x^{(i)}(\theta_{3kn+i}) \right) \right] \quad (4)$$

The role of CNOT gates is the introduction of entanglement in the system, which would otherwise be efficiently classically simulable. We select as the observable $O = \sigma_z^{(0)}$, which operates on a single qubit. Local observables like this one are less susceptible to the barren plateau problem than global ones, for example, $O = \bigotimes_{i=1}^n \sigma_z^{(i)}$ (as noted in [41]). The quantum neural network described in our investigation is pictured in Figure 4.

Training of the model To train models, we utilize a standard state-of-the-art gradient descent-based algorithm, ADAM. The Mean Squared Error (MSE) was selected as the loss function and error metric to evaluate the performances of the models in the regression tasks, as it is a standard error metric in supervised learning. MSE was selected as the loss function to train the networks because it is more sensitive to larger errors. Categorical Cross Entropy (CCE) was used as the loss function for the classification task instead, while Accuracy score was employed as error metric to assess the goodness of the classification. Given the output f of the model, the computation of its gradient ∇f , which is required to calculate the gradient of the loss function, is accomplished using the parameter-shift rule [57], since the commonly-used finite difference method $\nabla f(x; \theta) \approx (f(x; \theta) - f(x; \theta + \epsilon))/\epsilon$ is highly susceptible to hardware noise. The optimization hyper-parameters used are the learning rate, set to 0.1, and the number of training epochs, which was selected through empirical investigation (specifically, we carry out 150 training epochs to obtain the simulated results,

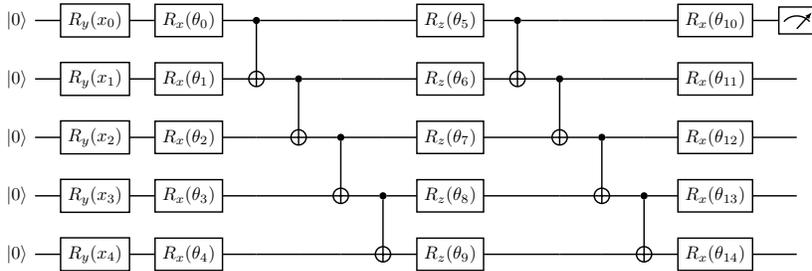


Figure 4: Quantum Neural Network used to classify the linear regression dataset, having 5 qubits and $\ell = 1$ layers. The rotational gates parameterized by the feature x_i form the feature map, while those parameterized via the θ s form the ansatz.

while for QPU-based results, we perform just 10 epochs due to technological constraints on current hardware).

Datasets We assess the performance of our approach using both synthetic and real-world datasets, across both regression and classification problems. The linear regression dataset is artificially generated with parametric control over the number of samples n , the dimensionality d , and the noise variance σ . It is procedurally generated by randomly sampling a weight vector w uniformly over $[-1, 1]^d$ such that the training set $\{(x^{(i)}, y^{(i)})\}_{i=1}^n$ is constructed with $x^{(i)}$ uniformly sampled from $[-1, 1]^d$, $y^{(i)} = w \cdot x^{(i)} + \epsilon^{(i)}$, and $\epsilon^{(i)}$ sampled from a normal distribution with zero mean and variance σ . In our case we have $n = 250$ (jointly the training and testing datasets), $d = 5$ and $\sigma = 0.1$. The other datasets involved in the experiments are the *Concrete Compressive Strength* dataset, the *Diabetes* dataset, and the *Wine* dataset. The first of these is a multivariate regression problem calculating the strength of the material based on its age and ingredients. The second is a multivariate regression problem correlating the biological and lifestyle characteristic of patients to their insulin levels. The third one is a multivariate, three-class classification problem investigating the geographic origin of wine samples from their chemical characteristics. All are freely available and open source. Table 2 summarizes the characteristics of these datasets. Every dataset is divided into 80% train samples and 20% test samples. Moreover, in a data preprocessing phase, raw data were scaled in the range $[-1, 1]$ to best suit the output of the quantum neural networks; the scaler was fitted using training data only. No other preprocessing technique, i.e. PCA, has been applied.

Implementation details Our implementation is written in Python3, and utilizes PennyLane as a framework to define and simulate quantum circuits, with the PennyLane-Qiskit plugin used to execute circuits on IBM Quantum devices

Dataset	Source	Nature	# Features	# Samples	Task
Linear	-	Synthetic	5	250	Regression
Concrete	UCI	Real-world	8	1030	Regression
Diabetes	Scikit-Learn	Real-world	10	442	Regression
Wine	UCI	Real-world	13	178	Classification

Table 2: Characteristics of the datasets analyzed. UCI stands for the open source *UCI Repository*. *Scikit-Learn* is an open-source software library for Python3. The number of features does not include the target.

via the Qiskit software stack. To improve simulation times, we employed the JAX linear algebra framework as the simulation backend. By using JAX, the quantum circuit can be just-in-time compiled to an intermediate representation called XLA, which can significantly speed up simulation times (by up to a factor of 10). Our simulations were run on a commercial computer with an AMD Ryzen 7 5800X (8-core CPU with a frequency of 3.80 GHz) and 64 GB of RAM. The experiments on the noise canceling properties of ensemble systems were conducted on the `ibm_lagos` quantum processing unit, which consists of 7 qubits arranged in the topology $\{(0, 1); (1, 2); (1, 3); (3, 4); (4, 5); (4, 6)\}$. The single-gate fidelity and CNOT fidelity of this QPU did not exceed $2.89e^{-4}$ and $8.63e^{-3}$, respectively (according to the latest calibration available).

4.2 Resource efficiency of quantum neural network ensembles

Besides performance, resource efficiency is a key argument for the utilization of quantum neural network ensembles. Efficiency can be measured by various metrics: for example, number of qubits, gates, parameters, and training samples required to achieve comparable performance.

To determine the potential savings in the number of qubits we here deploy the random subspace technique (also known as *attribute bagging* or *attribute bootstrap aggregation*). Our experiments (cf Figure 5) suggest a potential saving of 20% to 80% of the total qubit budget via this approach. However, such a saving is made at the cost of the ensemble as a whole having the potential for less rich class-discrimination behaviour, dependent on both the sampling required to achieve full feature coverage and the nature of the underlying data manifold. A positive consequence of reducing the number of qubits, though, is that each quantum circuit will have fewer gates and parameters, resulting in improved noise robustness on real hardware (i.e less decoherence, higher overall fidelity), as well as faster gradient calculation (individual gradient calculations require $P + 1$ quantum circuit evaluations for P parameters). This allows for a saving of the parameter budget of up to 75% in the indicated experimental regime, while the saving on gates corresponds proportionately (cf Figure 4). Savings for each dataset and ensemble technique are as depicted in Figure 5.

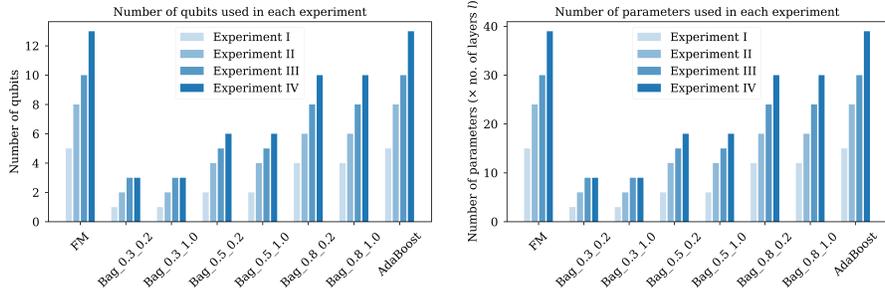


Figure 5: Number of qubits & parameters employed in individual experiments.

4.3 Simulated Domain Experiments

Initially, we evaluate our method in a simulated environment, one free of noise, such that the output estimation is infinitely precise. This differs significantly from execution on a NISQ quantum processing unit, which introduces various types of hardware error (such as decoherence and infidelity of operations) as well as sampling error caused via the measurement operation. We examine the performance of both the baseline models and ensemble systems in a scenario where the number of layers (i.e. quantum neural network depth) is gradually increased. To establish robustness to random initialization of parameters (that is, susceptibility to local minima effects), each simulation is repeated ten times.

4.3.1 Experiment I

The first experiment seeks to perform linear regression on a synthetic noisy 5-dimensional dataset. The function generating the targets is as follows: $y = w \cdot x + \epsilon$, where $x \in (-1, 1)^5 \subseteq \mathbb{R}^5$, $w \in \mathbb{R}^5$ is randomly generated from a uniform distribution having as support the range -1 to 1 , and ϵ is a Gaussian noise of mean zero and standard deviation 0.1 . The total number of samples composing this synthetic dataset is 250 . Each experimental data point instantiates a layer number, a number of bagged features, and a percentage of training data points available to the ensemble.

The results of the first experiment are indicated in Figure 6. Both FM and AdaBoost achieve the lowest MSE generalisation error of about 0.021 at 10 layers, reaching a performance plateau at 5 layers. The bagging models utilising 80% of the features are able to reach satisfactory results with 10 layers, which are only $0.03 - 0.05$ points higher than the error obtained by the best performing models. In general, it appears that quantum bagging models with a high number of features are able to generalize well on unseen data in this setting, even with only 20% of the training samples (unsurprisingly, the performance of bagging models with only 20% of training samples are worse than those of the counterparts using 100% of the training samples). Nevertheless, they still achieve remarkable results and show impressive generalization capabilities, confirming the effectiveness of

bagged quantum models in generalizing well with relatively little training data [64].

It is also notable that all of the bagging models have a lower MSE generalisation error as compared to FM and AdaBoost when the number of layers is low. In particular, with just 1 layer, all of the bagging models outperform FM and AdaBoost. However, as the number of layers increases, the performances of bagging models begin to plateau more rapidly than FM and Adaboost which, in contrast, continue their trend of decreasing error with increasing circuit depth. This is consistent with the notion that as base classifiers become expressive their risk of overfitting increases (i.e. they develop an intrinsically low bias). Adaboost, in particular, is known to be most effective in relation to weak, under-fitting base classifiers.

Finally, the decreasing error trend seen in the more complex bagging models as well as the FM and AdaBoost models is not visible in relation bagging with 30% of the features. We conjecture that since this bagging configuration utilises only 1 qubit, it cannot appropriately model the evolution of the quantum state with respect to the input. Hence, despite leveraging 10 different submodels of 1 qubit (i.e., one feature) each, the performance of bagging models with 30% of the features cannot improve as the number of layers increases (adding more layers in this case translates in performing rotations on the single qubit only, without the possibility of further CNOTs or other entangling gate operations). This result hence highlights the importance of entanglement in quantum neural network models as a means of improving performance.

4.3.2 Experiment II

The second experiment seeks to assess the performance of the respective ensemble techniques on the Concrete Compressive Strength dataset, which consists in 1030 samples of 8 features. The target value to predict in this regression case is hence the concrete compressive strength, measured in Megapascal (MPa), a highly nonlinear function of age and composition of the material.

The results of the regression experiment are in line with the findings of Experiment I, and are reported in Figure 7. FM, AdaBoost and the two bagging models applied in relation to 80% of features achieve comparable results at 10 layers, with the Bag..0.8-1.0 configuration obtaining the lowest MSE error, followed by Bag..0.8-0.2, FM and finally by AdaBoost. Also in this case, the differential between bagging models with 20% of samples and with 100% of samples is marginal, confirming the effectiveness of bagging quantum models in relation to reduced training dataset size. In contrast with Experiment I, bagging models having 30% of available features now have 2 qubits, and therefore demonstrate a relative improvement in test error when $l = 2$. However, their expressive power soon saturates and their error curves plateau.

In general, the generalization capability of bagging models decreases monotonically with the number of layers, in contrast to FM and AdaBoost. In fact, they exhibit episodes of overfitting when utilising 5 (and up to 7) layers, while bagging appears to be able to evade this outcome. This is again not surprising,

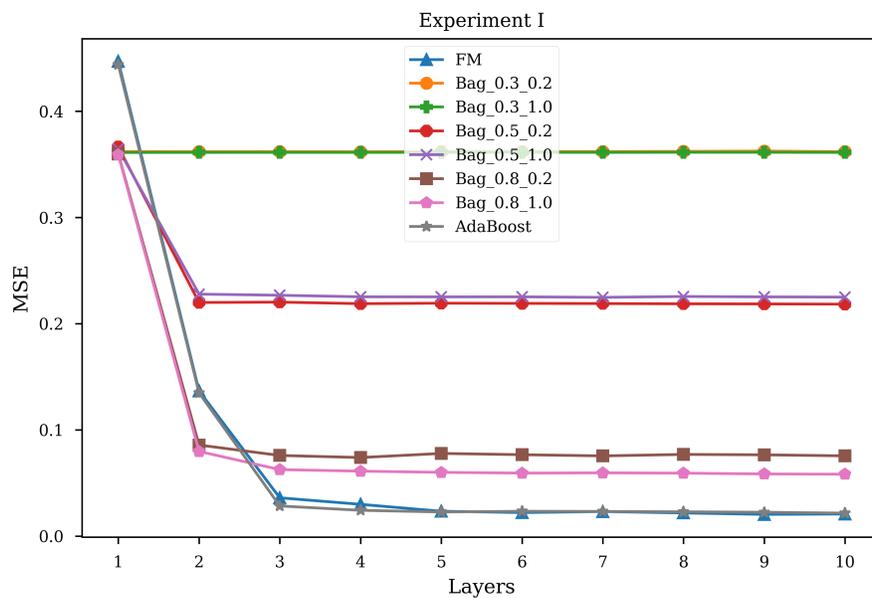


Figure 6: Evolution of MSE error with respect to the number of quantum neural network layers in Experiment I. Each experimental data point instantiates a layer number, a number of bagged features and a percentage of training data points available to the ensemble.

since AdaBoost is designed to reduce bias, while bagging ensembles are designed to reduce variance.

All of the bagging models analyzed still outperform FM and AdaBoost at a low number of layers, suggesting that they may be the right choice for implementation on NISQ devices, or else when there is any necessity of implementing low-depth quantum circuits. As in the first experiment, it is also of interest to note that all the bagging models with $l = 1$ here have very similar MSE values, while their performances vary as the number of layers increases. This may indicate that the MSE value reached at $l = 1$ is the optimal for that family of bagging models, given their expressibility. Moreover, a sharp decrease in MSE beyond the first layers would appear to be a common pattern, both with respect to the ensembles and the FM model. For example, at $l \geq 3$, the MSE error of FM and AdaBoost dramatically decrease, while bagging models with 50% of the features exhibit this trend between $l = 1$ and $l = 2$. (A future analysis of this topic might seek to exploit this characteristic in order to predict *a priori* how many layers one would need to attain an error level within a given bound).

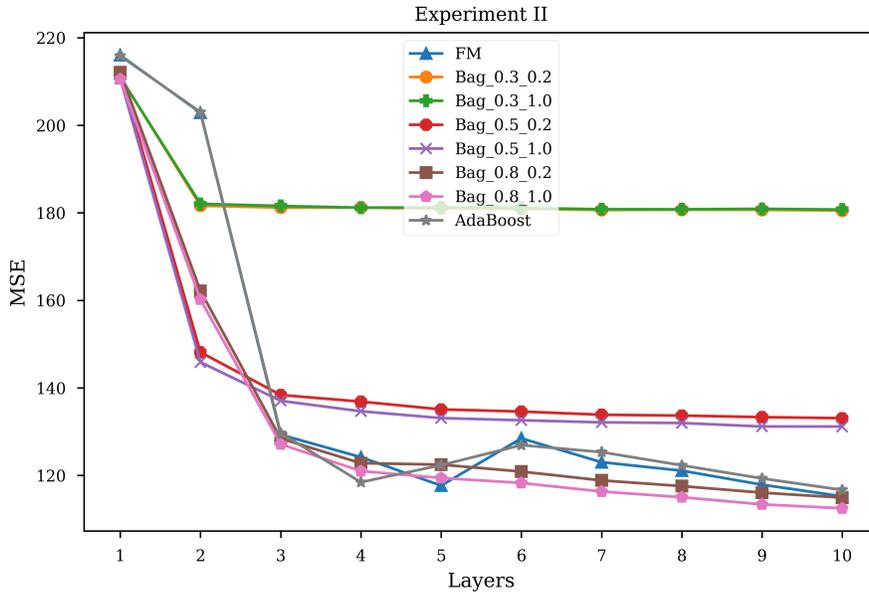


Figure 7: Evolution of MSE error with respect to the number of quantum neural network layers in Experiment II.

4.3.3 Experiment III

The dataset used in Experiment III is the reference Diabetes dataset from Scikit-learn, consisting of 10 numerical features, including age, sex, body mass index, blood serum measurements, and also a target variable, a quantitative measure

of disease progression one year after baseline. The dataset is composed of 442 instances and is often used for non-trivial regression analysis in ML.

Figure 8 illustrates the results of this experiment. The performance of the quantum models is notably different from those of the previous two experiments. It may be seen that the best performing models are the bagging models containing 80% of the features, while FM and AdaBoost achieve satisfactory results up to 6 layers, at which point their MSE begins to increase. At $l = 10$, every model has stabilized, however. Bag_.0.8_1.0 and Bag_.0.8_0.2 have an MSE of respectively 8.8% and 6.1% lower than that of FM. AdaBoost has an MSE comparable to the error of Bag_.0.3_1.0, being only 0.9% higher than FM. Bagging models with 50% of the features have surprisingly good results, better than those of FM and very close to bagging models with 80% of the features.

As in Experiment I and II, a very sharp MSE reduction between $l = 1$ and $l = 3$ is evident for all of the models. Less complex models like bagging with 30% and 50% of the features immediately reach a plateau, while the error curves for bagging with 80% of the features, FM and AdaBoost evolves as the number of parameters increases. Considering layer numbers between $l = 6$ and $l = 8$, it is clear that FM and AdaBoost overfit as the number of model parameters increases, and thus they perform poorly on test data. In particular, they overfit to such an extent that they almost reach the same performance level of the simplest bagging models with 30% of the features. The latter show no indication of overfitting however, in common with bagging models having 50% of the features. Bagging with 80% of the features shows light overfitting when $l > 6$, but still achieve the best results from among all of the tested algorithms.

The robustness of bagging models to overfitting with respect to AdaBoost and FM arises from their ability to reduce variance via averaging of decorrelated error across the predictions of each submodel. By contrast, when the number of layers is high, AdaBoost and FM utilise a model that is too complex and expressive for the underlying task, leading to overfitting. In concordance with Experiment II, this results suggests that attribute bagging is an effective solution to overfitting in the NISQ setting in common with that of the classical domain.

In addition, this experiment also highlights more markedly the discrepancy between the error level of bagging models with the same number of features but a distinct number of training samples. The difference between the MSE of the bagging model with 30% and 20% of samples and that with 100% of samples is now far more apparent, suggesting that when the variance of the dataset is very high, even bagging models require a sufficient threshold of training samples to perform well in the NISQ setting.

4.3.4 Experiment IV

For the classification task in Experiment IV, we used the reference UCI Wine dataset. It is a multi-class classification dataset corresponding to the results of a chemical analysis of wines grown within a specific region of Italy. It consists of 13 numerical features representing various chemical properties, such as alcohol, malic acid, and ash content, and a target variable indicating the class of the

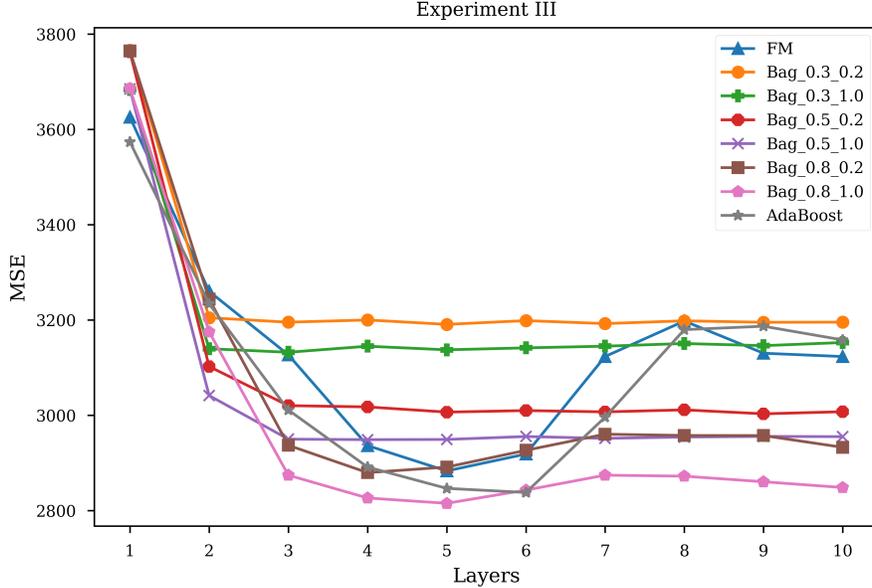


Figure 8: Evolution of MSE error with respect to the number of quantum neural network layers in Experiment III.

wine. The dataset has 178 samples and is a common baseline ML benchmark for low-parametric complexity classifiers.

Results from Experiment IV are reported in Figure 9. Although they cannot be directly compared to the previous results due to the intrinsically different nature of the problem, there are few comparative insights that can be gained from the respective plot of Accuracy curves. First, all the models except bagging with 30% of the features achieve the same accuracy score of 97.2% using 10 layers. The performances of Bag_.0.3_0.2 and Bag_.0.3_1.0 are still relatively strong, however, having an accuracy score of 94.2% and 96.9% respectively. Given the very low complexity of these two models, this is a striking result.

A further notable aspect of the Accuracy curves is that all ensemble models converge with far fewer layers than FM. In particular, they require 3 layers in order to reach a performance plateau on average, after which they saturate and the accuracy score reaches saturation as well. By contrast, FM struggles to achieve a comparable accuracy score, only achieving an accuracy greater than 90% when $l \geq 7$. This means that the ensemble models are able to learn and capture the complex relationships between the input features far more efficiently than FM, which requires a much deeper architecture to attain comparable results. This observation is particularly relevant when considering the implementation of these models on NISQ devices, where the number of qubits and the coherence time are severely limited.

Moreover, as expected, bagging models with 100% of the samples obtain a higher accuracy score than their counterparts with 20% of the features given the same number of layers. This suggests that using more training samples can improve the performance of ensemble models provided that the number of layers is low, as it allows them to better capture the underlying patterns of class discriminability in the data.

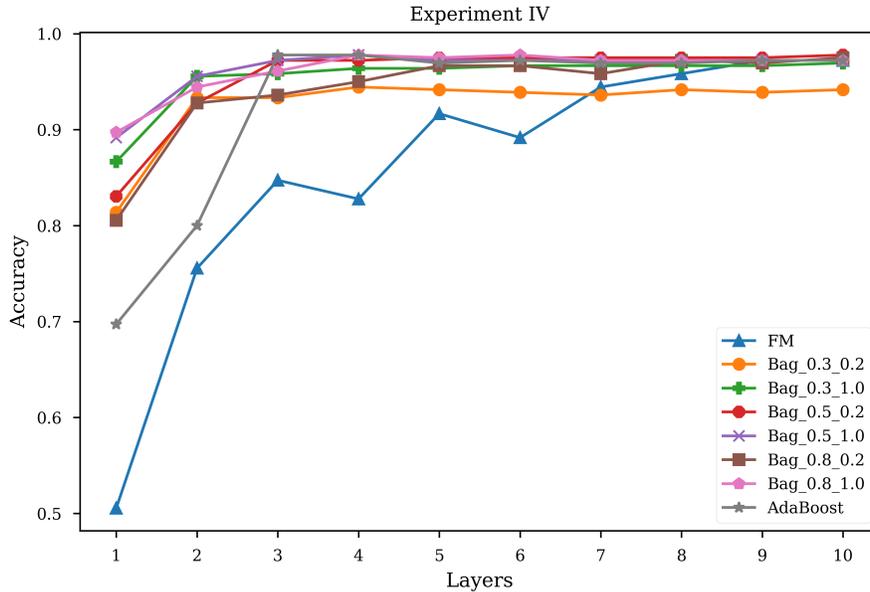


Figure 9: Evolution of Accuracy score with respect to quantum neural network depth in Experiment IV.

4.4 Experiments executed on superconducting-based QPU

For the real-hardware evaluation, we compare the performance of the baseline quantum neural network with the Bag_0.8_0.2 ensemble on the same synthetic linear regression dataset used in Experiment I. We selected the Bag_0.8_0.2 model as representative ensemble technique for its outstanding performance in the simulated experiments despite the low number of training samples. To ensure statistical validity, we repeat each experiment 10 times. However, due to technological constraints on real quantum hardware, we analyze only the linear dataset with a quantum neural network having a single layer.

Figure 10 presents the real-world experimental findings, which indicate that the bagging ensemble reduces the expected mean square error by one-third and the expected variance by half when executed on quantum hardware, compared to the baseline model. Such results demonstrate that the noise-canceling capabilities

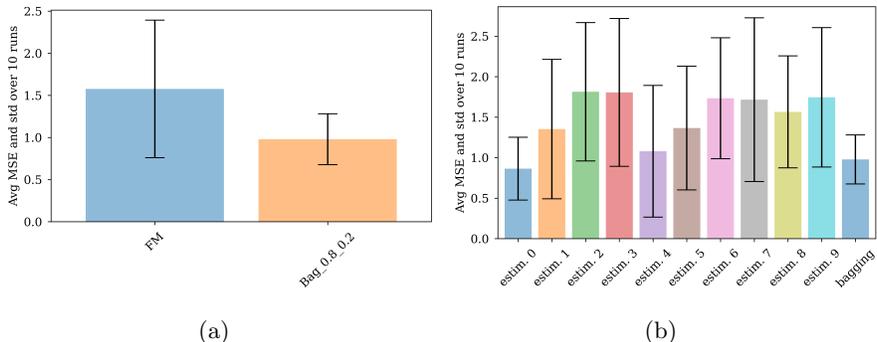


Figure 10: Comparison of average performance of the baseline model and the Bag_0.8_0.2 ensemble technique on IBM quantum hardware. (10a) shows the difference in terms of MSE over 10 executions. (10b) shows the performance of the bagging model with respect to its estimators.

of ensemble technique can be effectively exploited to work on NISQ devices in realistic settings. Additionally, the performance of the ten bagging models varied significantly, underlining the need to reinitialise the ensemble multiple times and validate it against a suitable validation dataset to ensure that the best model is selected.

5 Conclusion

We propose the use of ensemble techniques for practical implementation of quantum machine learning models on NISQ hardware. In particular, we justify the application of these techniques based on their capacity for significant reduction in resource usage, including in respect to the overall qubit, parameter, and gate budget, which is achieved via the random subspace (attribute bagging) technique. This resource-saving is especially crucial for noisy hardware, which is typically limited to a small number of qubits, being vulnerable to decoherence, noise, and operational errors. Consequently, the contribution of ensemble techniques may be seen as a form of quantum noise reduction.

To establish this, we evaluated and compared various configurations of bagging and boosting ensemble techniques on synthetic and real-world datasets, tested in both a simulated, noise-free environment and a superconducting-based QPU by IBM, and subtending a range of layer depths.

Our experimental findings showed that bagging ensembles can effectively train quantum neural network instances using fewer features and qubits, which leads to ensemble models with superior performance compared to the baseline model. Reducing the number of features in bagging models of quantum neural networks directly translates into a reduction in the number of qubits, that is a desirable characteristics for practical quantum applications. Ensembles of

quantum neural network can also help addressing some of the toughest challenges associated with noise and decoherence in NISQ devices, as well as to mitigate barren plateau effects. These can be key considerations in the development of quantum machine learning models, particularly when working with limited resources on modern quantum systems.

Moreover, bagging models were found to be extremely robust to overfitting, being able to effectively capture the underlying patterns in the data with high generalization ability. This makes them better suited for tasks where generalization is important, such as in real-world applications. However, it is important to notice that the effectiveness of bagging quantum models diminishes with a decrement in the number of features, which suggests that complex bagging models are still needed to obtain satisfactory results. Using only a subset of the features can reduce the computational complexity of the model and prevent overfitting, but it may also result in a loss of information and a decrease in performance. On the contrary, the number of training samples do not seem to have a deep impact on bagging quantum models, hence this bagging strategy may be used when executing quantum neural network instances on real hardware in order to deal with long waiting queues and job scheduling issues. In this regard, having a low number of training data leads to faster training procedures and quantum resource savings. The training of ensembles can also be done in parallel on multiple QPUs in a distributed learning fashion. Therefore, it is important to strike a balance between model complexity and performance to achieve the best possible outcomes.

Additionally, the fact that the bagging models outperform FM and AdaBoost at low number of layers suggests that the former models are better suited for low-depth quantum circuits, which have limited capacity and are prone to noise and errors. For quantum machine learning tasks with NISQ devices, using bagging models with a low number of layers may be a good strategy to achieve good generalization performance while minimizing the impact of noise and errors in the circuit.

Overall, our results suggest that ensembles of quantum neural network models can be a promising avenue for the development of practical quantum machine learning applications on NISQ devices, both from a performance and resource usage perspective. A careful evaluation of the trade-offs between model complexity, performance, quantum resources available and explainability may be necessary to make an informed decision.

In a future work, we plan to further investigate the relationship between ensembles and quantum noise, which is a key consideration when developing quantum neural network models. Our findings could potentially contribute to the development of more efficient and accurate quantum machine learning algorithms, which could have significant implications for real-world applications.

Acknowledgements

The contribution of M. Panella in this work was supported by the “NATIONAL CENTRE FOR HPC, BIG DATA AND QUANTUM COMPUTING” (CN1, Spoke 10) within the Italian “Piano Nazionale di Ripresa e Resilienza (PNRR)”, Mission 4 Component 2 Investment 1.4 funded by the European Union - NextGenerationEU - CN00000013 - CUP B83C22002940006. MG and SV are supported by CERN through CERN Quantum Technology Initiative. Access to the IBM Quantum Services was obtained through the IBM Quantum Hub at CERN. The views expressed are those of the authors and do not reflect the official policy or position of IBM and the IBM Q team. MI is part of the Gruppo Nazionale Calcolo Scientifico of “Istituto Nazionale di Alta Matematica Francesco Severi”. AM is supported by Foundation for Polish Science (FNP), IRAP project IC-TQT, contract no. 2018/MAB/5, co-financed by EU Smart Growth Operational Programme.

Declaration

Authors’ contributions

MI, MG, and AC had the initial idea, implemented the interface for executing experiments on the IBM QPUs, performed the experiments, and analyzed the data. MG, SV, DW, AM, and MP supervised the project. All authors contributed to the manuscript.

Availability of data and materials

The data and source code utilized in our study are freely accessible at <https://github.com/incud/Classical-ensemble-of-Quantum-Neural-Networks>. The procedural generation code for the Linear Regression dataset is also accessible at the same URL. In addition, the UCI Repository provides open access to Concrete and Wine datasets, which can be found at <https://archive.ics.uci.edu/ml/index.php>. The Diabetes dataset provided by Scikit-Learn is also freely available and included with the Python3 package.

References

- [1] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.
- [2] M Cerezo, Guillaume Verdon, Hsin-Yuan Huang, Lukasz Cincio, and Patrick J Coles. Challenges and opportunities in quantum machine learning. *Nature Computational Science*, 2(9):567–576, 2022.

- [3] Maria Schuld and Nathan Killoran. Is quantum advantage the right goal for quantum machine learning? *Prx Quantum*, 3(3):030101, 2022.
- [4] John Preskill. Quantum computing in the nisc era and beyond. *Quantum*, 2:79, 2018.
- [5] Amira Abbas, David Sutter, Christa Zoufal, Aurélien Lucchi, Alessio Figalli, and Stefan Woerner. The power of quantum neural networks. *Nature Computational Science*, 1(6):403–409, 2021.
- [6] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):1–6, 2018.
- [7] Zoe Holmes, Kunal Sharma, Marco Cerezo, and Patrick J Coles. Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum*, 3(1):010313, 2022.
- [8] Martin Larocca, Piotr Czarnik, Kunal Sharma, Gopikrishnan Muraleedharan, Patrick J Coles, and M Cerezo. Diagnosing barren plateaus with tools from quantum optimal control. *Quantum*, 6:824, 2022.
- [9] Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven, and Jarrod R McClean. Power of data in quantum machine learning. *Nature communications*, 12(1):2631, 2021.
- [10] Abdulkadir Canatar, Evan Peters, Cengiz Pehlevan, Stefan M Wild, and Ruslan Shaydulin. Bandwidth enables generalization in quantum kernel models. *arXiv preprint arXiv:2206.06686*, 2022.
- [11] Cha Zhang and Yunqian Ma. *Ensemble machine learning: methods and applications*. Springer US, New York, NY, 2012.
- [12] Nicolas De Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Cambridge Library Collection - Mathematics. Cambridge University Press, Cambridge, 2014.
- [13] Samet Oymak and Mahdi Soltanolkotabi. Toward moderate overparameterization: Global convergence guarantees for training shallow neural networks. *IEEE Journal on Selected Areas in Information Theory*, 1(1):84–105, 2020.
- [14] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.
- [15] Martin Larocca, Nathan Ju, Diego García-Martín, Patrick J Coles, and Marco Cerezo. Theory of overparameterization in quantum neural networks. *arXiv preprint arXiv:2109.11676*, 2021.

- [16] Junyu Liu, Francesco Tacchino, Jennifer R Glick, Liang Jiang, and Antonio Mezzacapo. Representation learning via quantum neural tangent kernels. *PRX Quantum*, 3(3):030323, 2022.
- [17] Massimiliano Incudini, Michele Grossi, Antonio Mandarino, Sofia Vallecorsa, Alessandra Di Pierro, and David Windridge. The quantum path kernel: a generalized quantum neural tangent kernel for deep quantum machine learning. *arXiv preprint arXiv:2212.11826*, 2022.
- [18] Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d’Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart. Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2):023401, 2020.
- [19] Angus Lowe, Matija Medvidović, Anthony Hayes, Lee J O’Riordan, Thomas R Bromley, Juan Miguel Arrazola, and Nathan Killoran. Fast quantum circuit cutting with randomized measurements. *arXiv preprint arXiv:2207.14734*, 2022.
- [20] Takeshi Yoshikawa, Tomoya Takanashi, and Hiromi Nakai. Quantum algorithm of the divide-and-conquer unitary coupled cluster method with a variational quantum eigensolver. *Journal of Chemical Theory and Computation*, 18(9):5360–5373, 2022.
- [21] Luca Asproni, Davide Caputo, Blanca Silva, Giovanni Fazzi, and Marco Magagnini. Accuracy and minor embedding in subqubo decomposition with fully connected large problems: a case study about the number partitioning problem. *Quantum Machine Intelligence*, 2(1):4, 2020.
- [22] Peng Zhang, Xingquan Zhu, Yong Shi, Li Guo, and Xindong Wu. Robust ensemble learning for mining noisy data streams. *Decision Support Systems*, 50(2):469–479, 2011.
- [23] Ryan LaRose, Andrea Mari, Sarah Kaiser, Peter J Karalekas, Andre A Alves, Piotr Czarnik, Mohamed El Mandouh, Max H Gordon, Yousef Hindy, Aaron Robertson, et al. Mitiq: A software package for error mitigation on noisy quantum computers. *Quantum*, 6:774, 2022.
- [24] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.
- [25] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. *Phys. Rev. A*, 98:032309, 09 2018.
- [26] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H Booth,

- et al. The variational quantum eigensolver: a review of methods and best practices. *Physics Reports*, 986:1–128, 2022.
- [27] Francesco Di Marcantonio, Massimiliano Incudini, Davide Tezza, and Michele Grossi. Quask—quantum advantage seeker with kernels. *arXiv preprint arXiv:2206.15284*, 2022.
- [28] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 17(9):1013–1017, 2021.
- [29] Marcello Benedetti, Mattia Fiorentini, and Michael Lubasch. Hardware-efficient variational quantum algorithms for time evolution. *Physical Review Research*, 3(3):033083, 2021.
- [30] Alexandre Choquette, Agustin Di Paolo, Panagiotis Kl Barkoutsos, David Sénéchal, Ivano Tavernelli, and Alexandre Blais. Quantum-optimal-control-inspired ansatz for variational quantum algorithms. *Physical Review Research*, 3(2):023092, 2021.
- [31] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.
- [32] Hrushikesh Patil, Yulun Wang, and Predrag S Krstić. Variational quantum linear solver with a dynamic ansatz. *Physical Review A*, 105(1):012423, 2022.
- [33] Maria Schuld, Alex Bocharov, Krysta M Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101(3):032308, 2020.
- [34] Fabio Valerio Massoli, Lucia Vadicamo, Giuseppe Amato, and Fabrizio Falchi. A leap among quantum computing and quantum neural networks: A survey. *ACM Comput. Surv.*, 55(5), 12 2022.
- [35] Vojtech Havlicek, Antonio D Corcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.
- [36] Antonio Macaluso, Luca Clissa, Stefano Lodi, and Claudio Sartori. A variational algorithm for quantum neural networks. In *International Conference on Computational Science*, pages 591–604. Springer, 2020.
- [37] Chen Zhao and Xiao-Shan Gao. Qdnn: deep neural networks with quantum layers. *Quantum Machine Intelligence*, 3(1):1–9, 2021.
- [38] Andrea Ceschini, Antonello Rosato, and Massimo Panella. Hybrid quantum-classical recurrent neural networks for time series prediction. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022.

- [39] Hanrui Wang, Jiaqi Gu, Yongshan Ding, Zirui Li, Frederic T Chong, David Z Pan, and Song Han. Quantumnat: quantum noise-aware training with noise injection, quantization and normalization. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 1–6, 2022.
- [40] Zhiding Liang, Zhepeng Wang, Junhuan Yang, Lei Yang, Yiyu Shi, and Weiwen Jiang. Can noise on qubits be learned in quantum neural network? a case study on quantumflow. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–7. IEEE, 2021.
- [41] Marco Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature communications*, 12(1):1–12, 2021.
- [42] Giovanni Seni and John F Elder. Ensemble methods in data mining: improving accuracy through combining predictions. *Synthesis lectures on data mining and knowledge discovery*, 2(1):1–126, 2010.
- [43] Naomi Altman and Martin Krzywinski. Ensemble methods: bagging and random forests. *Nature Methods*, 14(10):933–935, 2017.
- [44] Peter Bühlmann. Bagging, boosting and ensemble methods. In *Handbook of computational statistics*, pages 985–1022. Springer, Berlin, DE, 2012.
- [45] Ahmed Hamza Osman and Hani Moetque Abdullah Aljahdali. An effective of ensemble boosting learning method for breast cancer virtual screening using neural network model. *IEEE Access*, 8:39165–39174, 2020.
- [46] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.
- [47] Simon Berkhahn, Lothar Fuchs, and Insa Neuweiler. An ensemble neural network model for real-time prediction of urban floods. *Journal of hydrology*, 575:743–754, 2019.
- [48] Maria Schuld and Francesco Petruccione. Quantum ensembles of quantum classifiers. *Scientific reports*, 8(1):1–12, 2018.
- [49] Amira Abbas, Maria Schuld, and Francesco Petruccione. On quantum ensembles of quantum classifiers. *Quantum Machine Intelligence*, 2(1):1–8, 2020.
- [50] Daivid Leal, Tiago De Lima, and Adenilton J Da Silva. Training ensembles of quantum binary neural networks. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2021.
- [51] Antonio Macaluso, Luca Clissa, Stefano Lodi, and Claudio Sartori. Quantum ensemble for classification. *arXiv preprint arXiv:2007.01028*, 2020.

- [52] Samuel Stein, Nathan Wiebe, Yufei Ding, Peng Bo, Karol Kowalski, Nathan Baker, James Ang, and Ang Li. Eqc: ensembled quantum computing for variational quantum algorithms. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, pages 59–71, 2022.
- [53] David Windridge, Riccardo Mengoni, and Rajagopal Nagarajan. Quantum error-correcting output codes. *International Journal of Quantum Information*, 2018.
- [54] Ruiyang Qin, Zhiding Liang, Jinglei Cheng, Peter Kogge, and Yiyu Shi. Improving quantum classifier performance in nisq computers by voting strategy from ensemble learning. *arXiv preprint arXiv:2210.01656*, 2022.
- [55] Tanjung Krisnanda, Kevin Dini, Huawei Xu, Wouter Verstraelen, and Timothy CH Liew. Wisdom of crowds in quantum machine learning. *Physical Review Applied*, 19(3):034010, 2023.
- [56] Andrea Skolik, Stefano Mangini, Thomas Bäck, Chiara Macchiavello, and Vedran Dunjko. Robustness of quantum reinforcement learning under hardware errors. *EPJ Quantum Technology*, 10(1):1–43, 2023.
- [57] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331, 2019.
- [58] Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019.
- [59] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.
- [60] Harris Drucker. Improving regressors using boosting techniques. In *Icml*, volume 97, pages 107–115. Citeseer, 1997.
- [61] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.
- [62] Jonas Kübler, Simon Buchholz, and Bernhard Schölkopf. The inductive bias of quantum kernels. *Advances in Neural Information Processing Systems*, 34:12661–12673, 2021.
- [63] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran. Quantum embeddings for machine learning. *arXiv preprint arXiv:2001.03622*, 2020.
- [64] Matthias C Caro, Hsin-Yuan Huang, Marco Cerezo, Kunal Sharma, Andrew Sornborger, Lukasz Cincio, and Patrick J Coles. Generalization in quantum machine learning from few training data. *Nature communications*, 13(1):4919, 2022.

A Detailed plots

We provide some additional plots of the simulated experiments. In particular, we compare the different configurations of bagging and boosting techniques and their variance. Figure 11, 12, 13, 14 shows the results for the Linear, Concrete, Diabetes, and Wine datasets, respectively.

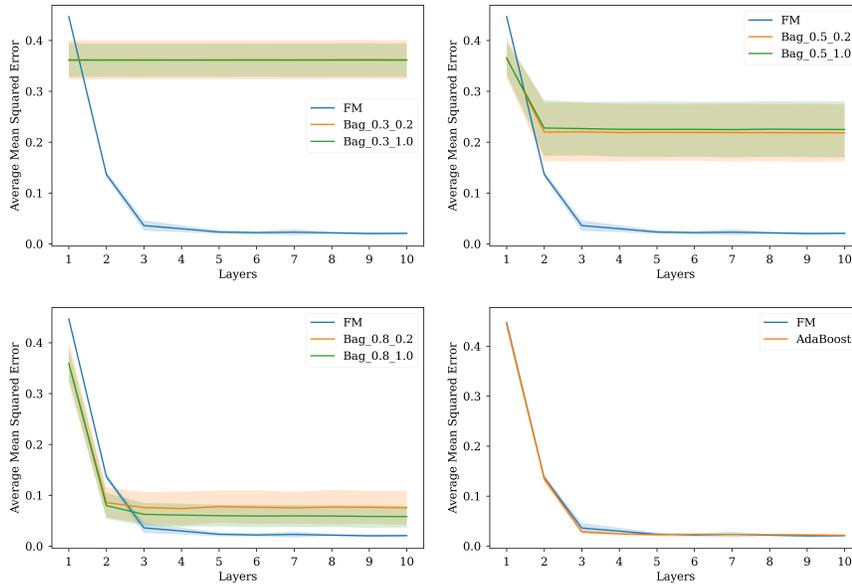


Figure 11: Comparison of the performance of the baseline model and ensemble systems on the Linear Regression dataset. It exhibits the MSE and standard deviation, with a semi-transparent area, of the ensemble schemes in comparison to the baseline models. The top-left image shows ensembles with Random Subspace at 30% of the features, top-right shows ensembles with Random Subspace at 50%, bottom-left displays ensembles with Random Subspace at 80%, and bottom-right illustrates AdaBoost.

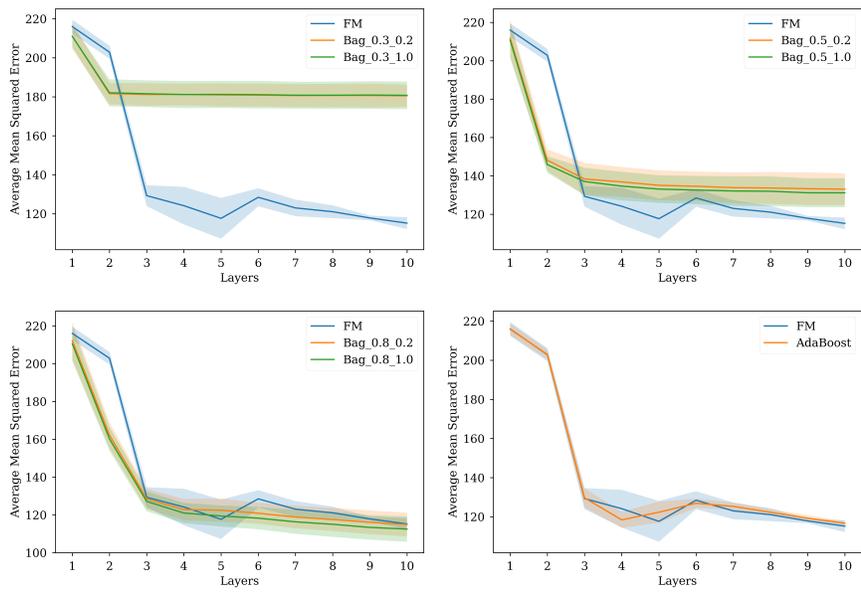


Figure 12: Comparison of the performance of the baseline model and ensemble systems on the Concrete Compressive Strength dataset. It exhibits the MSE and standard deviation, with a semi-transparent area, of the ensemble schemes in comparison to the baseline models. The top-left image shows ensembles with Random Subspace at 30% of the features, top-right shows ensembles with Random Subspace at 50%, bottom-left displays ensembles with Random Subspace at 80%, and bottom-right illustrates AdaBoost.

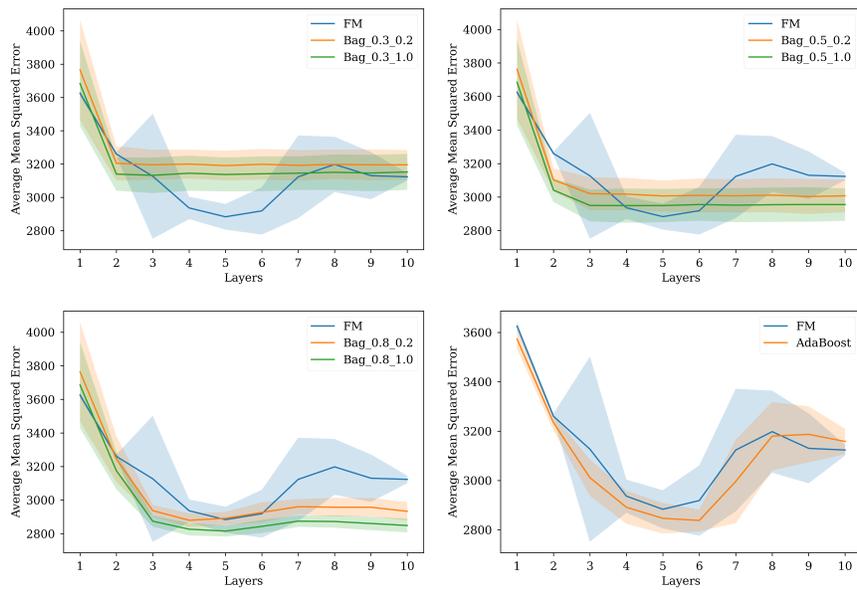


Figure 13: Comparison of the performance of the baseline model and ensemble systems on the Diabetes dataset. It exhibits the MSE and standard deviation, with a semi-transparent area, of the ensemble schemes in comparison to the baseline models. The top-left image shows ensembles with Random Subspace at 30% of the features, top-right shows ensembles with Random Subspace at 50%, bottom-left displays ensembles with Random Subspace at 80%, and bottom-right illustrates AdaBoost.

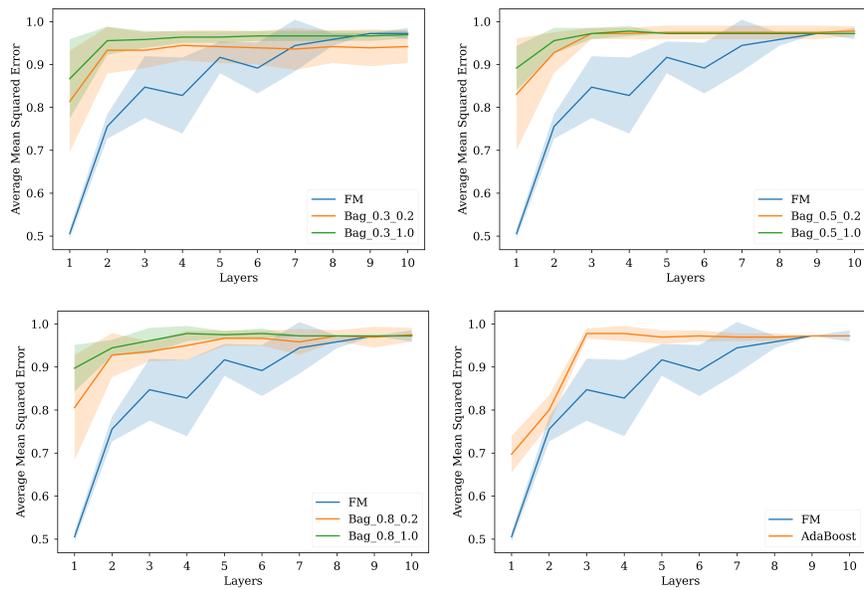


Figure 14: Comparison of the performance of the baseline model and ensemble systems on the Wine dataset. It exhibits the average accuracy and standard deviation, with a semi-transparent area, of the ensemble schemes in comparison to the baseline models. The top-left image shows ensembles with Random Subspace at 30% of the features, top-right shows ensembles with Random Subspace at 50%, bottom-left displays ensembles with Random Subspace at 80%, and bottom-right illustrates AdaBoost.