Efficient Parameter Optimisation for Quantum Kernel Alignment: A Sub-sampling Approach in Variational Training

M. Emre Sahin[1], Benjamin C. B. Symons[1], Pushpak Pati[2], Fayyaz Minhas[3], Declan Millar[4], Maria Gabrani[2], Stefano Mensa[1], and Jan Lukas Robertus[5,6]

[1]The Hartree Centre, STFC, Sci-Tech Daresbury, Warrington, WA4 4AD, United Kingdom

[2]IBM Research – Zurich, Säumerstrasse 4, Rüschlikon, CH–8803, Switzerland

[3]Tissue Image Analytics Centre, Warwick Cancer Research Centre, Department of Computer Science, University of Warwick, Coventry, CV4 7AL, United Kingdom

[4]IBM Research Europe - UK, Hursley, Winchester, SO21 2JN, United Kingdom

[5]Royal Brompton and Harefield Hosptials, Department of Histopathology, Guy's and St Thomas' NHS Foundation Trust, Sydney Street, London SW3 6NP United Kingdom

[6]National Heart and Lung Institute, Imperial College London, Guy Scadding Building, Dovehouse St, London SW3 6LY United Kingdom

arXiv:2401.02879v2 [quant-ph] 10 Oct 2024

Quantum machine learning with quantum kernels for classification problems is a growing area of research. Recently, quantum kernel alignment techniques that parameterise the kernel have been developed, allowing the kernel to be trained and therefore aligned with a specific dataset. While quantum kernel alignment is a promising technique, it has been hampered by considerable training costs because the full kernel matrix must be constructed at every training iteration. Addressing this challenge, we introduce a novel method that seeks to balance efficiency and performance. We present a sub-sampling training approach that uses a subset of the kernel matrix at each training step, thereby reducing the overall computational cost of the training. In this work, we apply the sub-sampling method to synthetic datasets and a real-world breast cancer dataset and demonstrate considerable reductions in the number of circuits required to train the quantum kernel while maintaining classification accuracy.

Stefano Mensa: stefano.mensa@stfc.ac.uk
Jan Lukas Robertus: j.robertus@imperial.ac.uk, SM and JLR share joint last authorship and contributed equally to this work. For all communications please address both authors.

1 Introduction

Kernel methods are a fundamental concept in Machine Learning (ML), with a rich history that spans several decades, revolutionising the field by enabling the effective handling of complex, nonlinear relationships in data [16]. The introduction of the "kernel trick" in the 1960s laid the foundation [2], leading to the development of Support Vector Machines (SVMs) in the 1990s [4], which popularized the concept. Over time, a diverse range of kernel functions have emerged, broadening their utility across many domains. Another milestone was the introduction of the concept of kernel alignment [9]. This method adjusts a kernel function in relation to specific data, thereby limiting the generalisation error [8, 35]. While facing challenges related to scalability [10], kernel methods continue to be relevant, coexisting with deep learning in solving complex problems.

The field of ML stands to benefit from quantum algorithms, offering the potential for improved scalability and accuracy in solving complex computational tasks [3, 22, 28, 30]. A particularly promising approach in this field is the utilization of quantum fidelity kernels, a quantum-enhanced version of kernel methods [15, 17, 22, 23, 29]. These kernels quantify the overlap between quantum states and leverage the high dimensionality of quantum systems to learn and

capture complex patterns in data. A few relevant practical applications of quantum machine learning with kernel methods can be seen in Refs. [12, 13, 24, 25, 36, 37].

Despite the promise of quantum kernel methods, there is scepticism about their practical application due to challenges associated with scalability. These challenges stem from the convergence of quantum kernel values which depends on factors such as data embedding expressibility, the number of qubits, global measurements, entanglement, and noise, which often demand increased measurements for effective training [7, 34]. In response to these challenges, several techniques have been introduced [17, 21, 32], which, along with bandwidth adjustments [6, 31], aim to enhance generalization in quantum kernels, particularly at higher qubit counts.

Quantum kernel alignment (QKA) adapts classical kernel alignment to utilize quantum kernels, enhancing the model's precision by aligning more effectively with core data patterns, as outlined in studies by [12, 14, 18]. Despite this advancement, the scaling in terms of the number of circuits required to align quantum kernels with QKA presents a considerable challenge. In the noiseless case the number of circuits at each training step scales quadratically with dataset size and in the presence of noise it has been shown to scale quartically [13], which can quickly become prohibitive for larger datasets [25, 34]. Addressing this complexity with respect to dataset size is crucial for enabling the scalability of QKA methods. One way to address the computational challenges of QKA is to adapt the PEGASOS algorithm [12]. In reference [12], they demonstrate that the complexity of this method for an ϵ-accurate classifier with a dataset size of m is $O(min\{m^2/\epsilon^6, 1/\epsilon^{10}\})$ as compared to $O(m^{4.67}/\epsilon^2)$ for standard quantum kernel alignment in the presence of finite sampling noise. Indeed the PEGASOS-based algorithm scales better than the standard approach with respect to m which makes it attractive for larger datasets but, it comes at the cost of worse scaling with respect to ϵ.

In this work, we aim to improve the scaling with respect to the dataset size of QKA methods by employing sub-sampling techniques during training. Specifically, we utilize a subset of the kernel matrix, termed the sub-kernel, in each training step. This approach aims to reduce the computational burden associated with training, while still effectively learning kernel parameters. As such, the prospective use of our sub-sampling approach opens up new avenues for scalable and efficient quantum machine learning algorithms.

2 Methodology

Classical kernel methods compute elements of a kernel matrix $K_{x,x'}$ using a kernel function $K(\boldsymbol{x}, \boldsymbol{x}')$ that measures the distance between two data points $\boldsymbol{x}$, $\boldsymbol{x}' \in X$. For certain choices of distance measure it is possible to use the kernel trick to express the kernel function as an inner product in some vector space V. If the kernel function can be written as a feature map $f : X \to V$, then V is known as the feature space and the kernel function can be written as an inner product on V, $K(\boldsymbol{x}, \boldsymbol{x}') = \langle f(\boldsymbol{x}), f(\boldsymbol{x}')\rangle_V$. We can see from this definition that the kernel function is a measure of similarity between data points that takes on values $[0, 1]$ e.g. if $x = x'$ then the inner product is one, whereas if the maps $f(x)$, $f(x')$ result in orthogonal vectors then the inner product is zero.

Quantum kernel methods can be defined analogously to classical kernel methods. A quantum feature map, $\phi(\boldsymbol{x})$, is used to transform classical data $\boldsymbol{x}$ into a corresponding quantum state $|\phi(\boldsymbol{x})\rangle$. The fidelity kernel between two classical data points, $\boldsymbol{x}$ and $\boldsymbol{x}'$, is computed using the overlap of the related quantum states $K(\boldsymbol{x}, \boldsymbol{x}') = |\langle\psi(\boldsymbol{x})|\psi(\boldsymbol{x}')\rangle|^2$, where $|\psi(\boldsymbol{x})\rangle = U_\phi(\boldsymbol{x})|0\rangle^{\otimes n}$ and $U_\phi(\boldsymbol{x})$ is the unitary that implements the quantum feature map. The quality of such kernels can be further improved by quantum kernel alignment techniques that parameterise the kernel and use a variational approach to optimize for the kernel for a specific dataset as suggested in Refs. [14, 18] (see Fig. 1). QKA adds a variational layer $U(\boldsymbol{\theta})$ to the kernel, where $\boldsymbol{\theta}$ are trainable parameters. The elements of the fidelity kernel matrix are computed in the same way, but now depend on $\boldsymbol{\theta}$ as shown in Equation 1,

$$K(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{\theta}) = |\langle\psi(\boldsymbol{\theta}, \boldsymbol{x})|\psi(\boldsymbol{\theta}, \boldsymbol{x}')\rangle|^2, \qquad (1)$$

where $|\psi(\boldsymbol{\theta}, \boldsymbol{x})\rangle = U_\phi(\boldsymbol{x})U(\boldsymbol{\theta})|0\rangle^{\otimes n}$. Note that, while we focus on kernels of the form shown in Equation (1), in principle our method generalises to other choices of kernel.

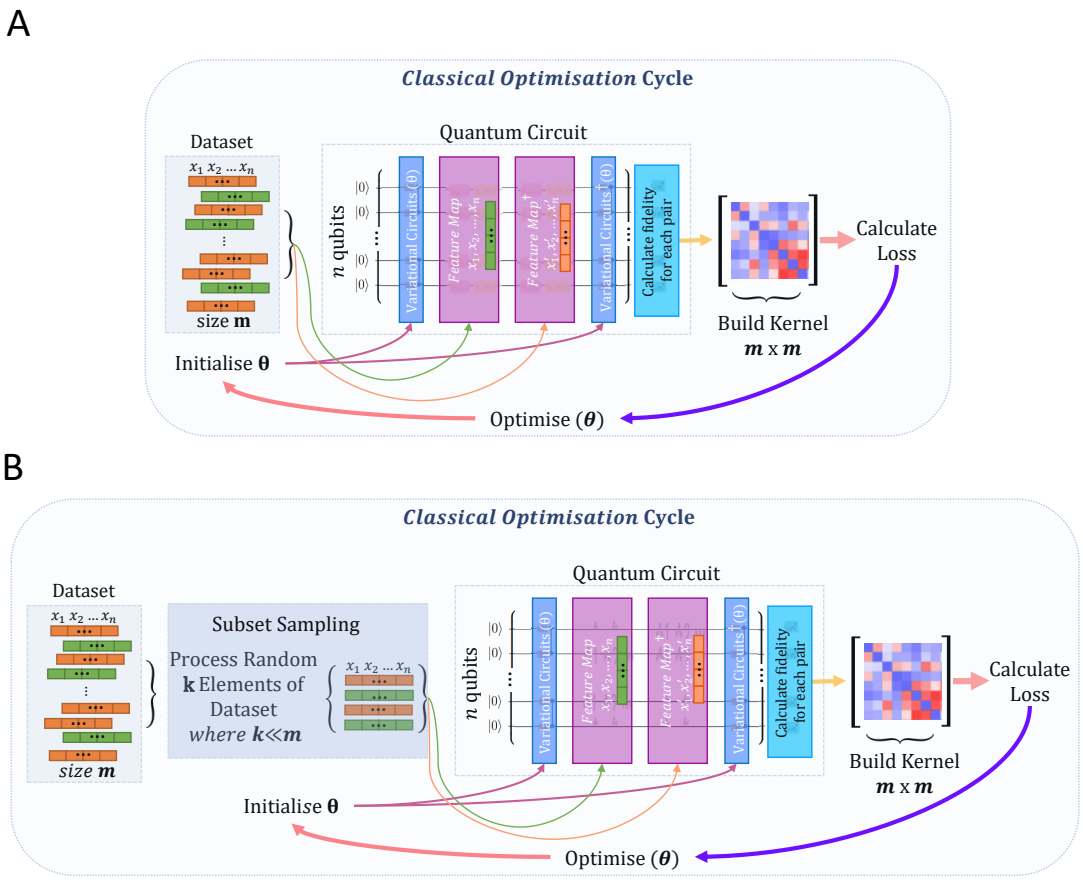

Figure 1: **A)** Quantum kernel alignment uses n qubits to encode the dataset into a feature map, estimate the kernel, and optimize parameters $\boldsymbol{\theta}$ until kernel separation improves. **B)** In the sub-sampling method, subsets with k elements of the dataset are similarly processed using n qubits. This cycle repeats with different subsets, optimizing $\boldsymbol{\theta}$ until the entire dataset is sampled, yielding an optimized kernel. This will end with the estimation of the full dataset, optimised kernel.

Similar to classical kernel alignment [9], the training of the variational quantum kernel can be seen as finding a kernel that minimizes the upper bound of the SVM generalization error. This is equivalent to minimising a loss function L,

$$L(\boldsymbol{\theta}, a) = \sum_i a_i - 0.5 \sum_{i,j} a_i a_j y_i y_j K(\boldsymbol{\theta}, x_i, x_j), \tag{2}$$

where a_i and a_j are the optimal Lagrange multipliers, and y are the labels of the binary classification problem. The hinge loss, as a function of the variational kernel $K(\boldsymbol{\theta})$, is not generally convex, hence the loss landscape may include local minima, complicating the training process.

While kernel alignment has been shown to improve the performance of kernel methods, it comes at a cost: for every iteration of the training process, the full kernel must be constructed which for a dataset with m points requires m^2 inner product evaluations. For quantum kernel alignment this translates to m^2 circuits per training iteration which, as the dataset grows, can become prohibitively costly. In this paper, we present a novel method that utilises a so-called sub-sampling approach to speed-up the model training process by reducing the number of circuits per training iteration.

The full kernel, $K_f(\boldsymbol{\theta})$ is defined on the full dataset D whereas the sub-kernel $K_s(\boldsymbol{\theta})$ is defined using a subset of the data $D_s \subset D$ that is randomly sampled from D. The sub-kernel is used during the variational optimisation phase in order to find optimal parameters $\boldsymbol{\theta}_{opt}$. Once obtained, the parameters $\boldsymbol{\theta}_{opt}$ are used to construct the full fidelity kernel $K_f(\boldsymbol{\theta}_{opt})$, which is then used by a Support Vector Classifier (SVC). The benefit of this approach is that fewer circuits are required to build a sub-kernel than the

full kernel, therefore training using a sub-kernel can speed-up the training process by a factor that is at worst constant and at best removes the dependence of scaling on m. The method, schematically represented in Fig. 1 B, can be summarised as follows:

- **Variational parameter initialisation:** the variational parameters $\boldsymbol{\theta}$ of the quantum circuit are initialised as $\boldsymbol{\theta}_{init}$, giving an initial state $|\psi(\boldsymbol{\theta}_{init})\rangle$.
- **Variational parameters and loss function optimisation loop:** execute the optimisation loop and for each iteration, execute the following sub-steps:
  - **Dataset subsampling:** initiate by randomly selecting a distinct subset of k data points from the full dataset and employ Equation (1) to construct a sub kernel, K_s. Note that each time a new subset is selected, it is composed of previously unselected points until the entire dataset is exhausted, at which point previously selected points may be used again.
  - **Loss function estimation:** compute the loss function L using K_s (Equation 2).
  - **Loss function optimisation:** perform an optimisation step on L to update the parameters $\boldsymbol{\theta}$.
- **Full kernel estimation:** utilise the optimised parameters $\boldsymbol{\theta}_{opt}$ to compute the full quantum kernel $K_f(\boldsymbol{\theta}_{opt})$.
- **Training of classifier:** implement the full kernel in an SVC model for binary classification.

When introducing sub-sampling into the optimisation process, instead of computing the loss $L_f = L(D)$ over the full dataset D, the loss is evaluated on a sub-sample of size k, $L_s = L(D_s)$. This introduces a sampling error ϵ_s. Assuming the sub-samples are i.i.d., the expected sampling error can be bounded by $\mathbb{E}[|\epsilon_s|] \leq O\left(\frac{\sigma_k}{\sqrt{s}}\right)$, where σ_k^2 is the variance of the distribution of losses for a sub-sample of size k and s is the number of samples. In the limit that the sub-sample size is equal to the full dataset, then the variance in the loss goes to zero. We cannot say anything general about σ_k as it will depend on the specifics of the dataset and the choice of k among other things. However, we expect that sub-sampling will introduce some non-zero variance that could be large. The error in the loss can be controlled by taking multiple sub-samples and averaging them in each step of the training. Nonetheless, the variance in the loss will likely result in more training iterations being required to reach some fixed convergence criterion, compared to the full kernel training method.

In the case of training with a full kernel, every iteration involves processing all m data points. The full kernel matrix contains m^2 entries which correspond to m^2 evaluations of the kernel function and therefore, assuming no noise, a query complexity of $O(m^2)$ per training iteration. In the presence of finite sampling noise, it has been shown that sampling overhead results in a considerably worse complexity of $O(m^{4.67}/\epsilon^2)$ per training iteration for a kernel that is ϵ-close to the infinite shot kernel [13]. For the remainder of this work, we consider the noise-free case but note that the speed-ups presented may in fact improve in the presence of noise.

Denoting the number of iterations required for convergence as T, the overall complexity for the full kernel training is $O(m^2T)$. On the other hand, with sub-kernel training, we only process a subset of the data points of size k at each iteration. Given T' training iterations and s sub-kernels per iteration, the overall complexity of the sub-sampling approach is $O(sk^2T')$. As discussed previously, it is likely that $T' > T$ meaning it is not guaranteed that the complexity is reduced. It is evident that the upper bound on T' is given by $T' < \frac{m^2}{sk^2}T$. Note that, while increasing s decreases the upper bound on T', it also decreases the loss error ϵ_s in each iteration which in turn is likely to decrease the actual value of T'. Assuming that $k << m$ and that $s < m^2/k^2$, then T' is allowed to be a potentially large multiple of T. This opens up the possibility for considerable reductions in run time when training using the sub-sampling approach. Note that, if s and k are chosen to be constants independent of m and T' does not scale with m, then the overall complexity of the training process is independent of m. While there are no *a priori* guarantees that T' will satisfy the bound given above, our results shown in the next section demonstrate significant

speed-ups with little to no loss of classification accuracy relative to the full kernel method for choices of s and k that are independent of m.

3 Results from Numerical Simulations and Discussion

In this section we demonstrate our novel method by applying it to several test cases, showcasing the reductions in optimised kernel estimation times as well as retained - or improved - classification accuracy. We start by describing the setup of our tests and then we present results using two synthetic and one real-world dataset as examples.

Our experiments involved two variational ansatze: Real Amplitudes (RA) and Hardware Efficient (HE) [11], often referred to as Efficient SU2 ansatze. We used three different optimisers including Gradient Descent (GD) [5], SPSA (Simultaneous Perturbation Stochastic Approximation) [33], ADAM (Adaptive Moment Estimation) [20]. We employed multiple commonly used ansatze and optimisers to test whether or not these choices impact the performance of the subsample method. We used different feature maps to encode the data according to the dataset and specify each in the relevant sections. To ensure fairness and reproducibility, the same initial variational parameters are used for every run. Unless stated otherwise, the results presented were obtained from statevector simulations using Qiskit [19]. The standard deviation of the outcomes through 10-fold cross-validation across all simulations was calculated. We maintained the same test and train subsets for all SVC methods to facilitate an accurate direct comparison between the different algorithms. All experiments were conducted five times, and the results shown are the averages over the values obtained.

In the results presented here, we use the number of queries as a figure of merit. This has the advantage of being a platform-independent metric. Here, a query refers to a single execution of a fidelity calculation i.e., a single circuit. By comparing the number of queries required to reach the optimal solution for each method, we can determine the relative speed-up one method has over another. Specifically, we define speed-up to be the ratio of the number of queries required for a full kernel compared to a given sub-kernel of size k.

3.1 Synthetic Datasets

3.1.1 Benchmarking for Second-order Pauli-Z evolution circuit

For the empirical evaluation of our method, we use a synthetic dataset that is constructed in line with the principles set out by Havlíček et al. [15], where the dataset is specifically designed to be fully separable through a second-order Pauli-Z evolution. The feature map transforms uniformly distributed vectors into quantum states, and labels are attributed based on the inner products of these quantum states. A separation gap of 0.2 is used to set the labeling threshold, ensuring a balanced yet challenging classification problem. The dataset comprises 96 training data points and 32 test data points, each represented by a 2-qubit quantum state.

In Fig. 2, we show the results obtained from statevector simulation using the SPSA optimiser with different ansatze, sub-samples sizes and the corresponding number of queries and relative speed-up compared to the full kernel evaluation. In this case, we set the maximum number of training iterations to 200 and if the optimiser reaches this point then we pick the point with the lowest loss. If the point with the lowest loss occurs before the final point, we consider this to be the stopping point of the run. In practice, a different stopping criterion such as a loss threshold could be chosen to avoid running more iterations than necessary. All 3 optimisers tested performed comparably, for the full list of results, see the Tables in Appendix B.

For the HE Ansatz results shown in the left panel of Fig. 2, the full kernel achieves an ROC AUC of 0.98 and an F1 score of 0.93. For all choices of k and s, the ROC AUC is slightly improved relative to the full kernel, with an average improvement of 0.7%. The F1 score achieved is identical to the full kernel in all cases apart from $k = 16$, $s = 1$ which sees an 8.2% drop in F1 score. For the RA Ansatz results shown in the right panel of Fig. 2, the full kernel achieves the same ROC AUC and F1 score as for the HE Ansatz. The parameters $k = 8$ and $s = 8$ are the only choice for which the ROC AUC and F1 decrease relative to the full kernel (by 0.51% and 3.5% respectively). For all other parameter

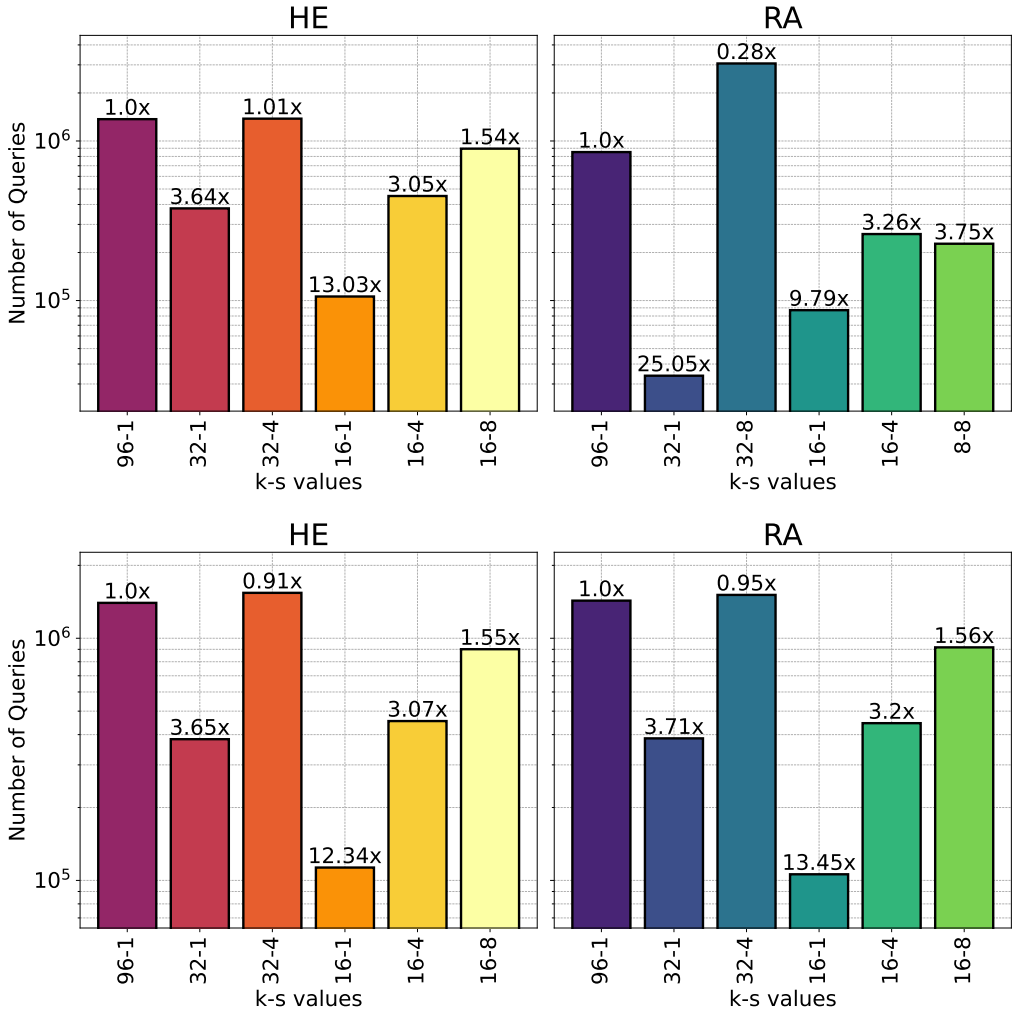

Figure 2: Total number of queries and speed-up relative to full kernel for the top 12 results using the SPSA optimizer, grouped by Hardware Efficient (HE) ansatze and Real Amplitudes (RA) ansatze. **Upper panels:** the range of ROC AUC values is 0.975-0.990 and the F1 scores range from 0.856-0.931 on the ZZ dataset (see Table 3 for full results). **Lower panels:** all ROC AUC values are 1.0 and the F1 values range from 0.969-1.0 on the Coset dataset (see Table 7 for full results). In both panels, k is the sub-kernel size, and s is the number of samples. Results are ordered by descending k.

choices shown the F1 score is identical to the full kernel score and the ROC AUC increases by 0.51-1.02%.

The results demonstrate that the sub-sampling method can achieve similar performance using fewer circuits. The number of queries shown in Figure 2 is the total number of queries, i.e. the number of queries per iteration multiplied by the number of iterations. We therefore see that, despite the sub-sampling method requiring more iterations to converge the loss function compared to the full kernel method (see Appendix D Figs 6 - 9), the total number of queries is still reduced considerably in almost all cases, resulting in up to x25 speed-up (reduction in number of queries). The only case for which there is a slowdown is for the RA ansatz when $k = 32$ and $s = 8$. This is perhaps unsurprising given that $m = 96$ and so with these parameters the ratio m^2/sk^2 is relatively close to 1. Furthermore, the accuracy of the classification is not deteriorated by the sub-sampling method, showing comparable or even improved accuracy compared to the calculation of the full kernel ($k = 96$ and $s = 1$ for both HE and RA ansatze).

We also performed a simple hardware experiment for this test case with the SPSA optimiser that can be seen in Appendix C Table 2. The experiments were performed on IBM Quantum hardware, Nairobi, using 2 qubits with 100 shots

and no error mitigation. While we cannot draw general conclusions from these experiments, it is interesting to note that the speed-ups are realised even when measuring run time rather than queries. Furthermore, while the performance in terms of ROC AUC and F1 is lower in all cases (full and sub-kernel), we observe several cases where the sub-sampling method improves classification accuracy relative to the full kernel. This is some initial evidence that the method works on real hardware in the presence of noise.

3.1.2 Learning Problem Labeling Cosets with Error (LCE)

The second synthetic dataset we tested was the learning problem Labeling Cosets with Error (LCE) dataset, which was crafted to evaluate the accuracy of QML algorithms. The dataset was generated with a 27-qubit superconducting quantum processor [14] which is split by 96 training points and 32 test points, each a 7-qubit quantum state defined by Euler angles. These angles specify elements of two separate cosets, forming the basis for binary classification. We apply the same feature map as used in [14].

Similarly to Section 3.1.1, we present the results obtained using the SPSA optimiser in Figure 2 (see Appendix E Tables for the full set of results). Also in this case, we observe over a $\times 10$ speed-up, with the number of queries reduced as we reduce k. For both the HE and RA ansatzes in Fig. 2, the full kernel ROC AUC and F1 score are 1. For all choices of k and s the ROC AUC remains at 1. All F1 scores are also 1 apart from the choice of $k = 16$, $s = 8$ which sees a small reduction to 0.969. We therefore see that the classification accuracy of the method is, once again, not affected by the method and agrees with our previous discussion in Section 3.1.1.

3.2 Real-World Dataset Experiment: Computational Pathology Breast Cancer Dataset

In the previous two subsections, we highlighted the strengths of our method with two synthetic datasets, showing that it's possible to train a quantum kernel faster, and without compromising accuracy using a sub-sampling approach. To further substantiate these benefits, we tested our method on a real-world application where high classification accuracy is non-negotiable - computational pathology for detecting and subtyping cancer. We chose this as an example dataset because it is a classification problem that is known to be classically challenging and where small improvements in accuracy can have a significant real-world impact. While the aim of this work is not to improve accuracy over the state-of-the-art classical methods, we believe the speed-ups presented in this work may help to facilitate future progress towards this goal.

Accurate cancer detection and subtyping (i.e. the process of categorising a cancer into a more specific group based on its characteristics) [26], involves analyzing complex data from diagnostic images using Graph Neural Networks (GNNs). Typically the GNN is used to extract features that are then passed to a classifier [1] (see Fig. 3). These classifiers have been shown to have high accuracy in classifying invasive vs non-invasive cancers but lower accuracy when classifying subtypes [27].

In this work, we take the features extracted from a GNN using the framework as described in [27] on breast cancer (BRACS) data. In the dataset, there are three distinct pathological benign lesions including Usual Ductal Hyperplasia (UDH), Flat Epithelial Atypia (FEA) and Atypical Ductal Hyperplasia (ADH), and malignant lesions including Ductal Carcinoma In Situ (DCIS) and Invasive Carcinoma (IC) besides Normal breast tissue. The subtypes are divided into two complex categories. The first category compares ADH and FEA to DCIS (1566 training points, 275 test points). In the second category Normal, and UDH are compared to ADH, FEA and DCIS (2797 Training points, 524 Test points).

In our tests, we used a 10-qubit feature map inspired by an instantaneous quantum polynomial (IQP) kernel [15, 17] with an optimal bandwidth parameter $c_{opt} \approx 2/n$ [6, 31] to reduce effects of exponential concentration quantum kernels [34].

$$U_{IQP}(x) = U_Z(x) \otimes H^n U_Z(x) \otimes H^n,$$
$$U_Z(x) = \exp\left(c \sum_{j=1}^{n} x_j Z_j + c^2 \sum_{j,j'=1}^{n} x_j x_{j'} Z_j Z_{j'}\right), \tag{3}$$

where c is the bandwidth, H and Z denote the Hadamard and Pauli Z gates respectively.

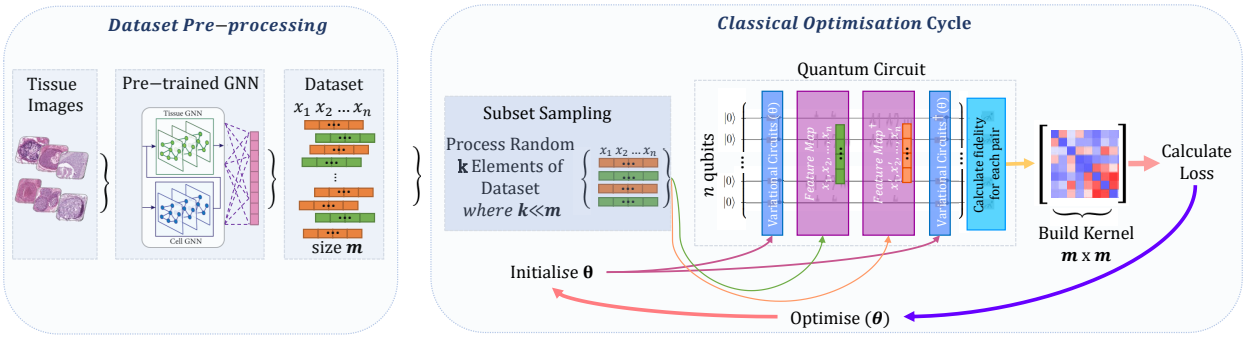

Figure 3: Combined GNN-extracted [27] hierarchical cellular features with our sub-sampling method to optimise the creation of a quantum kernel used to classify subtypes of cancer. The GNN-extracted embeddings are used to create the dataset that will be sampled using the methodology discussed in this work. The output of the workflow is a fully variationally optimised quantum kernel.

It's important to note that the results (shown in Fig. 4) are influenced by the inherent limitations of this feature map. Given that the goal of this work is to demonstrate the performance of the sub-sampling method, we chose this feature map not because it is optimal but simply because it is commonly used and readily available. We believe that it may be possible to improve classification accuracy beyond the results presented here by, for example, exploring different feature maps.

The results highlighted in Fig. 4 are collected to show the performance of the sub-sampling method comparing all the optimisers at once. In the same Figure, we also make the comparison between the classification accuracy of a GNN, classical kernels, PEGASOS kernel alignment [12] and a quantum fidelity kernel without kernel alignment. In the case of the sub-kernel method applied to the real-world problem of cancer subtyping, we observe similar trends to those observed for the synthetic datasets discussed above. The sub-sampling method does not degrade the classification performance,(top panels in Fig. 4). It is also evident that the quantum kernel with QKA outperforms both the quantum fidelity kernel without kernel alignment and PEGASOS kernel alignment in terms of classification accuracy (cyan line and blue dotted line in Fig. 4). The comparison of the results for the breast cancer data set between the sub-kernel method and the classical GNN approach indicates that the sub-kernel method can achieve comparable results to the classical method. While the initial findings are promising, they should be interpreted as indicative rather than conclusive.

The advantages of employing the sub-sampling method to accelerate kernel training are also clear (bottom panels in Fig. 4). Given the substantial size of the full kernels in these cases (ranging from ∼1600 to ∼2800 data points), the reduction in queries is even more significant than previously observed, with some instances requiring three orders of magnitude fewer queries. When a kernel of size $m = 2796$ is reduced to a sub-kernel of size $k = 8$ and $s = 1$, the number of queries is reduced by a factor of 1.6×10^3. These results provide some evidence that it is possible to choose the parameters s and k as constants independent of m. In the case that $m = 1566$, we find that choosing $k = 16$ and $s = 1$ gives a worst case reduction in ROC AUC and F1 score of 1.77% and 1.02% respectively. On average ROC AUC increases by 1.16% and the F1 score decreases by 0.1% (see Appendix E Table X). Increasing the full kernel size to $m = 2796$ gives similarly good results for the same parameters, $k = 16$ and $s = 1$. The worst case reduction in ROC and F1 score is 1.18% and 0.36% respectively. On average the ROC AUC increases by 0.61% the F1 score increases by 0.02% (see Appendix E Table XII). While we can't generalise from these two data points, the fact that we observe that k and s do not need to grow with m in this case is an encouraging sign that the sub-kernel method can provide highly efficient scaling, potentially inde-

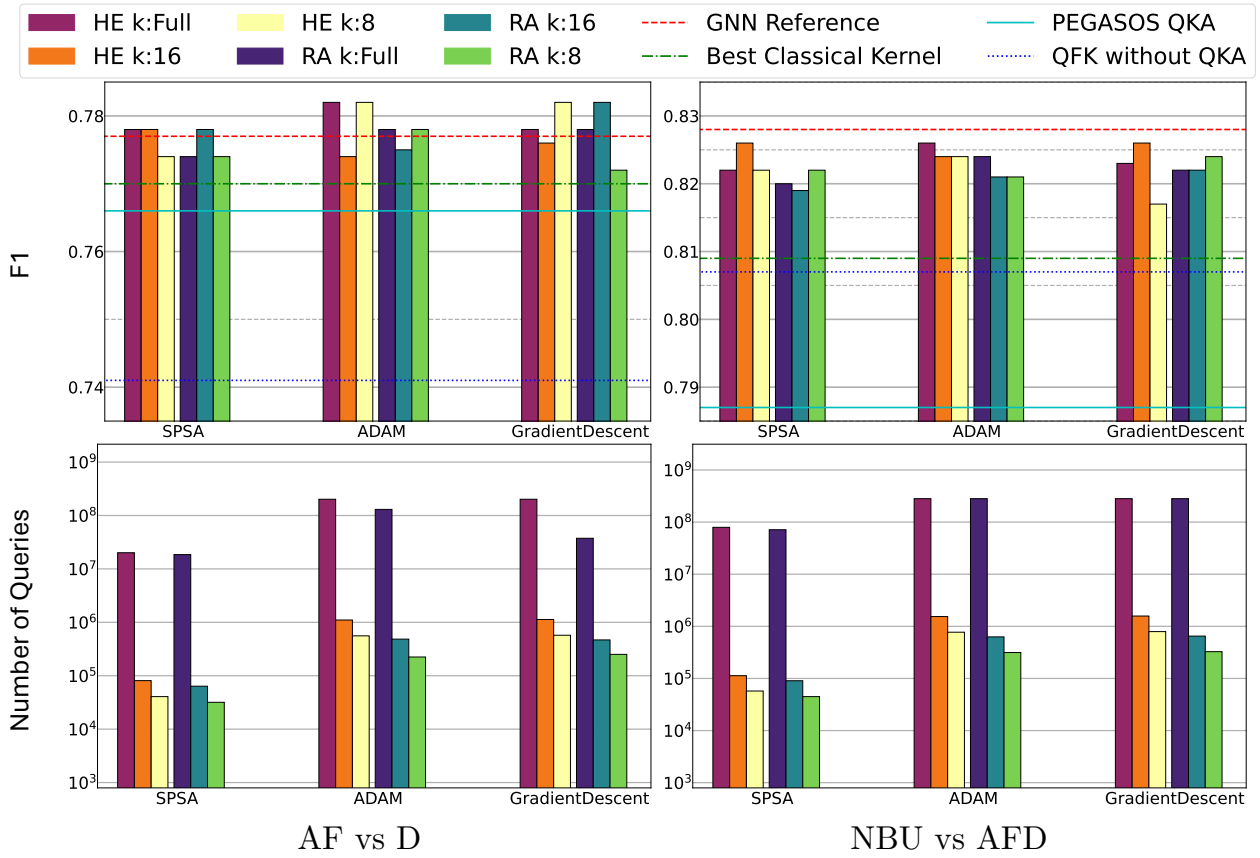

Figure 4: Top left and right panels, F1 scores obtained with our method, grouped by optimiser (ADAM, GD and SPSA), using the *full dataset* and for the classification of AF vs D and NBU vs AFD cancer subtypes. Quantum fidelity kernel, best classical kernel and GNN reference values are represented as horizontal lines. Bottom left and right panels: total number of queries for each method, grouped by optimiser. The best results are obtained after a parameter grid search as described in A, and full results for PEGASOS QKA is given in B

pendent of m.

4 Conclusion

This study proposes a novel approach for implementing quantum kernel alignment for Support Vector Classification (SVC), introducing the use of sub-sampling during the quantum training phase. QKA is a method that is capable of improving the performance of quantum kernels. However, QKA comes at a significant cost due to the need to train the kernel. This cost has, until now, undermined the practical usefulness of quantum kernel alignment. However, our results have demonstrated that the sub-sampling method is a promising approach that makes QKA more viable in the near-term by considerably reducing number of circuits required to train the kernel. Importantly we have observed that this reduction in training cost does not necessarily come at the cost of classification accuracy.

While the sub-sampling method has the potential to speed-up the training process, there are no *a priori* guarantees that this will be the case. The actual reduction (or increase) in queries will depend on the choice of sub-kernel size, number of samples per training iteration and the specifics of the dataset. Nonetheless, in practice we have found that, in the cases tested, the speed-up is remarkably robust across different parameters, ansatzes and classical optimisers. The vast majority of instances tested resulted in reductions in number of circuits, even when testing on a real-world dataset. In fact the real-world dataset gave the best results with reductions of up to 3 orders of magnitude. Furthermore, the sub-sampling

method introduces a level of stochastic optimisation which could lessen the severity of errors. This may explain why we observe improvements in classification accuracy in several cases when using the sub-sampling method. However, the stochastic nature of the training phase may also cause stability and reproducibility issues.

Acknowledgments

This work was supported by the Hartree National Centre for Digital Innovation, a collaboration between the Science and Technology Facilities Council and IBM. We thank Stefan Woerner, Francesco Tacchino and Ivano Tavernelli for their valuable inputs. The project concept and research proposal was created and developed by Imperial College London and the Royal Brompton and Harefield Hospitals in collaboration with IBM. We also thank Voica Ana Maria Radescu, Michele Grossi, Chris Burton, Frederik Flöther for their valuable support and input to the initial project development and concept.

References

[1] David Ahmedt-Aristizabal, Mohammad Ali Armin, Simon Denman, Clinton Fookes, and Lars Petersson. A survey on graph-based deep learning for computational histopathology. *Computerized Medical Imaging and Graphics*, 95:102027, 2022.

[2] MA Aiserman, Emmanuil M Braverman, and Lev I Rozonoer. Theoretical foundations of the potential function method in pattern recognition. *Avtomat. i Telemeh*, 25 (6):917–936, 1964.

[3] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[4] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[5] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[6] Abdulkadir Canatar, Evan Peters, Cengiz Pehlevan, Stefan M Wild, and Ruslan Shaydulin. Bandwidth enables generalization in quantum kernel models. *arXiv:2206.06686*, 2022.

[7] 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.

[8] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13(1): 795–828, 2012.

[9] Nello Cristianini, John Shawe-Taylor, André Elisseeff, and Jaz Kandola. On kernel-target alignment. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

[10] Bo Dai, Bo Xie, Niao He, Yingyu Liang, Anant Raj, Maria-Florina F Balcan, and Le Song. Scalable kernel methods via doubly stochastic gradients. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, volume 27. Curran Associates, Inc., 2014. ISBN 9781510800410.

[11] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv:1411.4028*, 2014.

[12] Gian Gentinetta, David Sutter, Christa Zoufal, Bryce Fuller, and Stefan Woerner. Quantum kernel alignment with stochastic gradient descent. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 01, pages 256–262, 2023.

[13] Gian Gentinetta, Arne Thomsen, David Sutter, and Stefan Woerner. The complexity of quantum support vector machines. *Quantum*, 8:1225, 2024.

[14] Jennifer R Glick, Tanvi P Gujarati, Antonio D Corcoles, Youngseok Kim, Abhinav Kandala, Jay M Gambetta, and Kristan Temme. Covariant quantum kernels for data with group structure. *Nature Physics*, 20(3):479–483, 2024.

[15] Vojtěch Havlíček, Antonio D Córcoles, 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.

[16] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36 (3):1171 – 1220, 2008.

[17] 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.

[18] Thomas Hubregtsen, David Wierichs, Elies Gil-Fuster, Peter-Jan HS Derks, Paul K Faehrmann, and Johannes Jakob Meyer. Training quantum embedding kernels on near-term quantum computers. *Physical Review A*, 106(4):042431, 2022.

[19] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D Nation, Lev S Bishop, Andrew W Cross, et al. Quantum computing with qiskit. *arXiv:2405.08810*, 2024.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[21] Jonas M. Kübler, Simon Buchholz, and Bernhard Schölkopf. The inductive bias of quantum kernels. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9781713845393.

[22] 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.

[23] Riccardo Mengoni and Alessandra Di Pierro. Kernel methods in quantum machine learning. *Quantum Machine Intelligence*, 1(3-4): 65–71, 2019.

[24] Stefano Mensa, Emre Sahin, Francesco Tacchino, Panagiotis Kl Barkoutsos, and Ivano Tavernelli. Quantum machine learning framework for virtual screening in drug discovery: a prospective quantum advantage. *Machine Learning: Science and Technology*, 4(1):015023, 2023.

[25] Shungo Miyabe, Brian Quanz, Noriaki Shimada, Abhijit Mitra, Takahiro Yamamoto, Vladimir Rastunkov, Dimitris Alevras, Mekena Metcalf, Daniel JM King, Mohammad Mamouei, et al. Quantum multiple kernel learning in financial classification tasks. *arXiv:2312.00260*, 2023.

[26] Karen O'Leary. Precision medicine for advanced breast cancer. *Nature Medicine*, 2022.

[27] Pushpak Pati, Guillaume Jaume, Antonio Foncubierta-Rodriguez, Florinda Feroce, Anna Maria Anniciello, Giosue Scognamiglio, Nadia Brancati, Maryse Fiche, Estelle Dubruc, Daniel Riccio, et al. Hierarchical graph representations in digital pathology. *Medical image analysis*, 75:102264, 2022.

[28] Diego Ristè, Marcus P Da Silva, Colm A Ryan, Andrew W Cross, Antonio D Córcoles, John A Smolin, Jay M Gambetta, Jerry M Chow, and Blake R Johnson. Demonstration of quantum advantage in machine learning. *npj Quantum Information*, 3(1):16, 2017.

[29] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Physical Review Letters*, 122(4): 040504, 2019.

[30] Maria Schuld and Francesco Petruccione. *Machine learning with quantum computers*. Springer, 2021.

[31] Ruslan Shaydulin and Stefan M Wild. Importance of kernel bandwidth in quantum machine learning. *Physical Review A*, 106 (4):042407, 2022.

[32] Norihito Shirai, Kenji Kubo, Kosuke Mitarai, and Keisuke Fujii. Quantum tangent kernel. *arXiv:2111.02951*, 2021.

[33] James C Spall. Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Transactions on Aerospace and Electronic Systems*, 34(3): 817–823, 1998.

[34] Supanut Thanasilp, Samson Wang, M Cerezo, and Zoë Holmes. Exponential concentration in quantum kernel methods. *Nature Communications*, 15(1): 5200, jun 2024.

[35] Tinghua Wang, Dongyan Zhao, and Shengfeng Tian. An overview of kernel alignment and its applications. *Artificial Intelligence Review*, 43:179–192, 2015.

[36] Sau Lan Wu, Shaojun Sun, Wen Guan, Chen Zhou, Jay Chan, Chi Lung Cheng, Tuan Pham, Yan Qian, Alex Zeng Wang, Rui Zhang, et al. Application of quantum machine learning using the quantum kernel algorithm on high energy physics analysis at the lhc. *Physical Review Research*, 3(3): 033221, 2021.

[37] Yusen Wu, Bujiao Wu, Jingbo Wang, and Xiao Yuan. Quantum phase recognition via quantum kernel methods. *Quantum*, 7:981, 2023.

A Details of Experiments

In our numerical experiments, we utilized specific software and dataset settings to ensure reproducibility and precision. The software versions employed were Qiskit version 0.44, Qiskit Machine Learning version 0.6, and scikit-learn Version 1.3.0. For the Synthetic Benchmarking Dataset, the settings were: SPSA with 200 maximum iterations, a learning rate of 0.01, and a perturbation of 0.05; ADAM with 200 maximum iterations, a tolerance of 1e-06, and a learning rate of 0.01; Gradient Descent with 200 maximum iterations, a learning rate of 0.01, a tolerance of 1e-07, and no perturbation. For the COSET Dataset and the Computational Pathology Breast Cancer Dataset, maximum iterations were increased to 400 for all algorithms, keeping other parameters constant.

Using a validation set, which is split from the training set, a grid-search is performed over C and γ parameters as well as kernel type (Poly, RFB, Linear), and the best parameters and kernel type are then used to find best Classical Kernel to train the SVC algorithm. Similarly, a grid-search is performed over different C values for the obtained quantum kernels using the same validation set for both QKA and PEGASOS QKA algorithms.

B Results of PEGASOS QKA

Table 1: Reference Values for NBU vs AFD and AF vs D. Best results of PEGASOS QKA [12] from the grid search as described in Appendix A

NBU vs AFD	
Number of Steps	**F1 Score**
10	0.602
100	0.706
1,000	0.765
10,000	0.787
100,000	0.780
AF vs D	
Number of Steps	**F1 Score**
10	0.737
100	0.766
1,000	0.763
10,000	0.760
100,000	0.760

C Results for Hardware run of Synthetic Dataset with SPSA Optimizer

The hardware used in this work is IBM's *ibm_nairobi* with Falcon r5.11H processor and 7 qubits. For qubit connectivity refer to Fig. 5. Qiskit options were set as *initial_layout=[1,2]*, *transpile=3* and *resillience-level=0* with 100 shots.

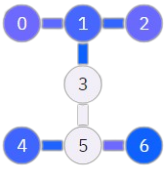

Figure 5: Schematic of the IBM quantum hardware *ibm_nairobi* device taken at the time of the hardware experiments.

Table 2: It has been only run once. Hardware results for SPSA Optimizer on the synthetic benchmarking dataset in section 2.1.1. k is the sub-kernel sample size, and s is the number of samples. The F1 and ROC scores are computed on the full, optimised kernel. Results are ordered by descending k and grouped by ansatze. The runtime is calculated as the total hardware execution time inside an IBM Quantum Session.

Ansatz	k	s	ROC AUC	F1	Runtime (s)
HE	96	1	0.85	0.721	20155
HE	32	1	0.904	0.906	2819
HE	32	8	0.94	0.889	16331
HE	16	4	0.995	0.953	3590
HE	4	8	0.992	0.953	2422
RA	96	1	0.86	0.8	19453
RA	32	1	0.807	0.694	2864
RA	32	8	0.859	0.766	15506
RA	16	4	0.891	0.812	3851

D Loss Graphs

The magnitudes of the loss values computed for different settings can vary considerably due to the use of different kernel sizes, leading to significantly different scales of loss. Direct comparisons of these values are not informative because of this scale variation. To facilitate meaningful comparisons, min-max scaling is applied to each loss graph in Figs. 6 - 8, standardizing the range of values. The purpose of these plots is to demonstrate that, as expected, the sub-sampling method does result in the loss converging at a slower rate compared to the full kernel method.

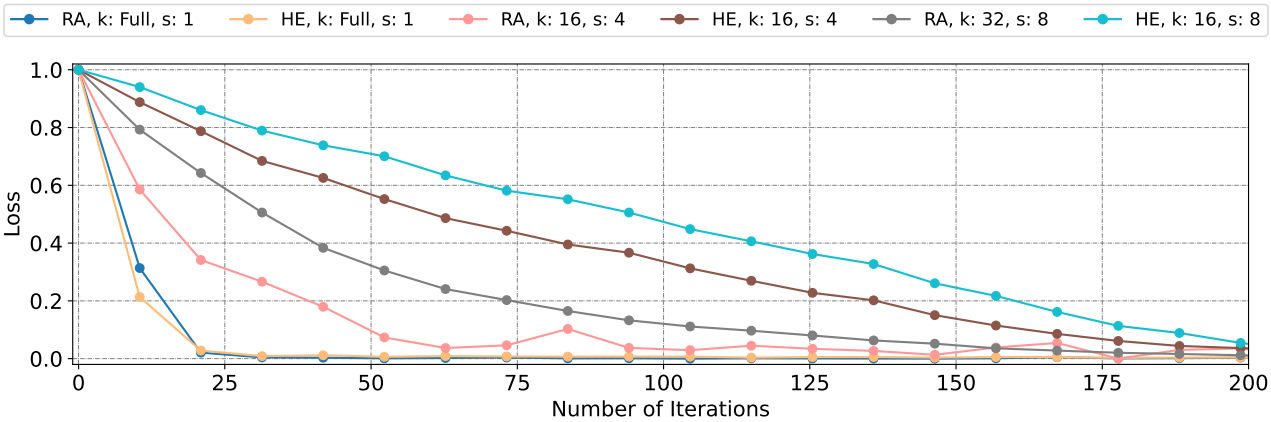

Figure 6: Min-Max normalised loss function for SPSA optimiser on the synthetic benchmarking dataset in section 2.1.1

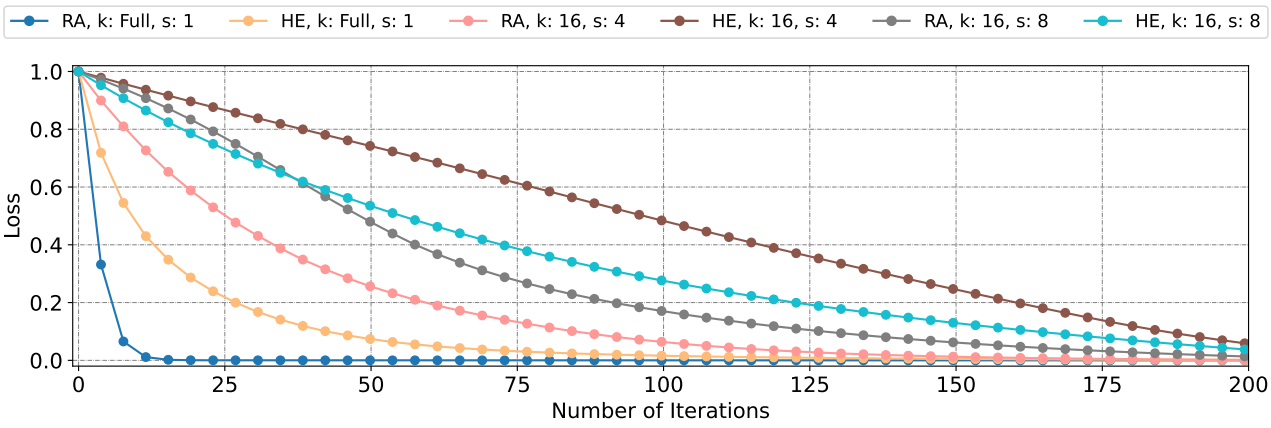

Figure 7: Min-Max normalised loss function for GD optimiser on the synthetic benchmarking dataset in the section 2.1.1

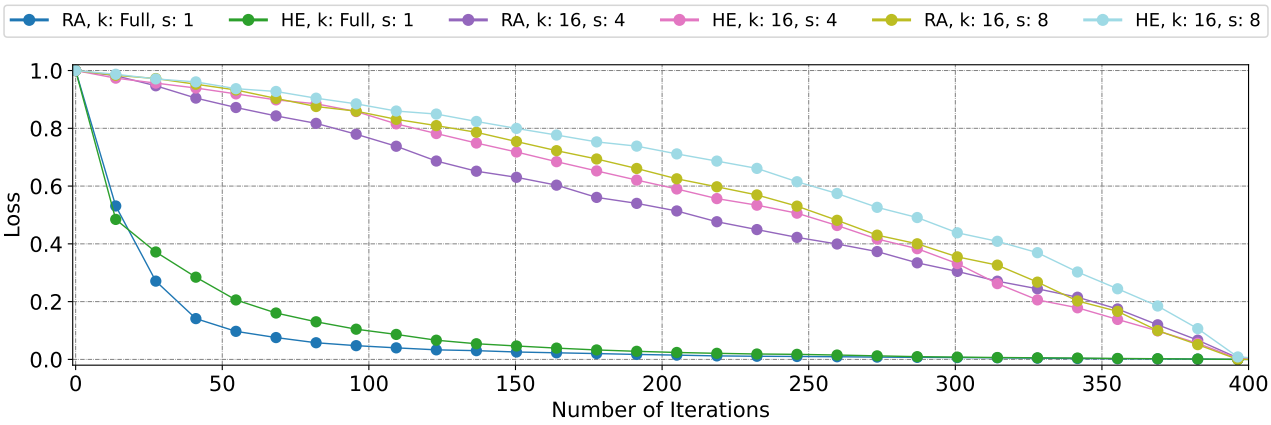

Figure 8: Min-Max normalised loss function for SPSA optimiser on COSETS problem.

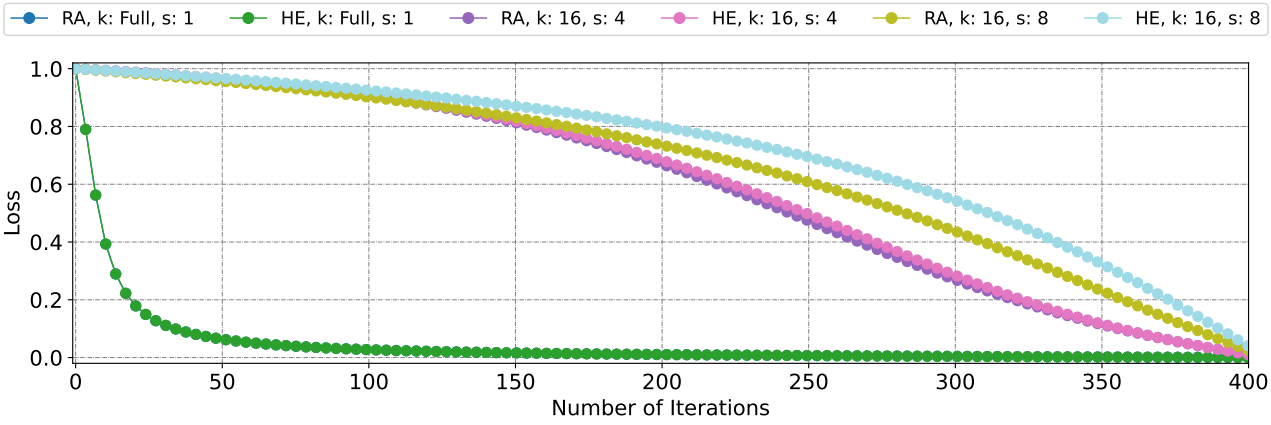

Figure 9: Min-Max normalised loss function for GD optimiser on COSETS problem.

E Numerical Results

Table 3: Results for SPSA Optimizer on the synthetic benchmarking dataset in section 2.1.1. k is the sub-kernel sample size, and s is the number of samples. The F1 and ROC scores are computed on the full, optimised kernel. Results are ordered by descending k and grouped by ansatze. Full kernels ($k = m$) are highlighted in bold.

Ansatz	k	s	ROC AUC	F1	Queries	Speed-up
HE	**96**	**1**	**0.98**	**0.9312**	**1377696**	**1**
HE	32	1	0.99	0.9312	378848	3.64
HE	32	4	0.985	0.9312	1380320	1
HE	32	8	0.98	0.9312	3125216	0.44
HE	16	1	0.985	0.8561	105712	13.03
HE	16	4	0.99	0.9312	452080	3.05
HE	16	8	0.985	0.9312	894960	1.54
HE	8	1	0.98	0.9312	14808	93.04
HE	8	4	0.985	0.9312	108536	12.69
HE	8	8	0.985	0.9312	213752	6.45
HE	4	1	0.96	0.7407	3012	457.40
HE	4	4	0.965	0.8667	26908	51.20
HE	4	8	1	0.9312	54460	25.30
RA	**96**	**1**	**0.98**	**0.9312**	**852384**	**1**
RA	32	1	0.985	0.9312	33760	25.25
RA	32	4	0.985	0.9312	1558496	0.55
RA	32	8	0.990	0.9312	3063776	0.28
RA	16	1	0.985	0.9312	87024	9.79
RA	16	4	0.985	0.9312	260784	3.26
RA	16	8	0.89	0.8244	919536	0.93
RA	8	1	0.98	0.8624	27960	30.49
RA	8	4	0.97	0.8986	101496	8.4
RA	8	8	0.975	0.8986	227064	3.75
RA	4	1	0.905	0.8244	4972	171.44
RA	4	4	0.97	0.8986	28636	29.77
RA	4	8	0.865	0.7634	56700	15.03

Table 4: Results for ADAM Optimizer on the synthetic benchmarking dataset in section 2.1.1. k is the sub-kernel sample size, and s is the number of samples. The F1 and ROC scores are computed on the full, optimised kernel. Results are ordered by descending k and grouped by ansatze. Full kernels ($k = m$) are highlighted in bold.

Ansatz	k	s	ROC AUC	F1	Queries	Speed-up
HE	**96**	**1**	**0.985**	**0.9312**	**6114720**	**1**
HE	32	1	0.97	0.7407	2157024	2.83
HE	32	4	0.915	0.8244	3012576	2.03
HE	32	8	0.96	0.8624	491488	12.44
HE	16	1	0.95	0.8561	111472	54.85
HE	16	4	0.955	0.8624	1893872	3.23
HE	16	8	0.98	0.9312	902128	6.78
HE	8	1	0.98	0.9312	344	17775.35
HE	8	4	0.975	0.9662	689400	8.87
HE	8	8	0.975	0.8624	158456	38.59
HE	4	1	0.915	0.7937	60	101912
HE	4	4	0.985	0.9312	148220	41.25
HE	4	8	0.955	0.8624	344956	17.73
RA	**96**	**1**	**0.91**	**0.5333**	**350112**	**1**
RA	32	1	0.94	0.8986	56800	6.16
RA	32	4	0.965	0.8244	88032	3.98
RA	32	8	0.905	0.8	9957344	0.04
RA	16	1	0.87	0.6936	66416	5.27
RA	16	4	0.985	0.8624	68080	5.14
RA	16	8	0.975	0.8624	984048	0.36
RA	8	1	0.85	0.8000	184	1902.78
RA	8	4	0.915	0.8148	383736	0.91
RA	8	8	0.87	0.7937	768248	0.46
RA	4	1	0.895	0.8244	116	3018.21
RA	4	4	0.97	0.9662	32060	10.92
RA	4	8	0.93	0.8624	172732	2.03

Table 5: Results for GD Optimizer on the synthetic benchmarking dataset in section 2.1.1. k is the sub-kernel sample size, and s is the number of samples. The F1 and ROC scores are computed on the full, optimised kernel. Results are ordered by descending k and grouped by ansatze. Full k are highlighted in bold.

Ansatz	k	s	ROC AUC	F1	Queries	Speed-up
HE	**96**	**1**	**0.98**	**0.9312**	**6437280**	**1**
HE	32	1	0.98	0.9312	1163232	5.53
HE	32	4	0.985	0.9312	6948832	0.93
HE	32	8	0.98	0.9312	13897696	0.46
HE	16	1	0.98	0.9312	323952	19.87
HE	16	4	0.99	0.8947	2047984	3.14
HE	16	8	0.98	0.9312	4088816	1.57
HE	8	1	0.97	0.8624	2520	2554.48
HE	8	4	0.965	0.8947	511992	12.57
HE	8	8	0.965	0.8986	1023224	6.29
HE	4	1	0.975	0.8624	31876	201.95
HE	4	4	0.975	0.8561	68668	93.74
HE	4	8	0.85	0.7542	189180	34.03
RA	**96**	**1**	**0.98**	**0.9312**	**2386848**	**1**
RA	32	1	0.915	0.831	1036768	2.3
RA	32	4	0.985	0.9312	4171744	0.57
RA	32	8	0.85	0.7333	8347616	0.29
RA	16	1	0.865	0.5333	2416	987.93
RA	16	4	0.985	0.9312	1226736	1.95
RA	16	8	0.99	0.9312	2455536	0.97
RA	8	1	0.945	0.8667	2520	947.16
RA	8	4	0.93	0.831	192376	12.41
RA	8	8	0.97	0.8624	614392	3.88
RA	4	1	0.81	0.7249	6244	382.26
RA	4	4	0.985	0.8624	76604	31.16
RA	4	8	0.915	0.7634	13564	175.97

Table 6: Results for SPSA Optimizer on the COSET dataset in section 2.1.2. k is the sub-kernel sample size, and s is the number of samples. The F1 and ROC scores are computed on the full, optimised kernel. Results are ordered by descending k and grouped by ansatze. Full kernels ($k = m$) are highlighted in bold.

Ansatz	k	s	ROC AUC	F1	Queries	Speed-up
HE	**96**	**1**	**1**	**1**	**1396128**	**1**
HE	32	1	1	1	382944	3.65
HE	32	4	1	1	1542112	0.91
HE	32	8	1	1	3125216	0.45
HE	16	1	1	1	113136	12.34
HE	16	4	1	1	455152	3.07
HE	16	8	1	0.9687	901104	1.55
HE	8	1	0.9569	0.811	11736	118.96
HE	8	4	0.8510	0.75	96504	14.47
HE	8	8	0.7765	0.75	228600	6.11
HE	4	1	0.7412	0.75	1668	837.01
HE	4	4	0.7137	0.75	18940	73.71
HE	4	8	0.6941	0.7179	35452	39.38
RA	**96**	**1**	**1**	**1**	**1428384**	**1**
RA	32	1	1	1	385504	3.71
RA	32	4	1	1	1507296	0.95
RA	32	8	1	1	3084256	0.46
RA	16	1	1	1	106224	13.45
RA	16	4	1	1	446448	3.20
RA	16	8	1	0.9687	917488	1.56
RA	8	1	0.6157	0.6161	22040	64.81
RA	8	4	0.9255	0.811	115064	12.41
RA	8	8	0.9137	0.811	229880	6.21
RA	4	1	0.7882	0.75	2972	480.61
RA	4	4	0.6980	0.7179	27516	51.91
RA	4	8	0.7686	0.75	33660	42.44

Table 7: Results for ADAM Optimizer on the COSET dataset in section 2.1.2. k is the sub-kernel sample size, and s is the number of samples. The F1 and ROC scores are computed on the full, optimised kernel. Results are ordered by descending k and grouped by ansatze. Full kernels ($k = m$) are highlighted in bold.

Ansatz	k	s	ROC AUC	F1	Queries	Speed-up
HE	**96**	**1**	**1**	**1**	**7741344**	**1**
HE	32	1	0.9137	0.811	2088928	3.71
HE	32	4	0.9765	0.811	8173536	0.95
HE	32	8	0.9804	0.9375	5533664	1.40
HE	16	1	0.9176	0.8125	614512	12.60
HE	16	4	0.7176	0.7179	1090544	7.10
HE	16	8	0.8118	0.75	4916208	1.57
HE	8	1	1	1	153272	50.51
HE	8	4	0.6941	0.7179	18936	408.82
HE	8	8	0.7255	0.75	146424	52.87
HE	4	1	0.9843	0.906	38268	202.29
HE	4	4	0.9725	0.874	153500	50.43
HE	4	8	1	1	306812	25.23
RA	**96**	**1**	**1**	**1**	**7741344**	**1**
RA	32	1	0.7647	0.75	1883616	4.11
RA	32	4	0.8980	0.811	8355808	0.93
RA	32	8	0.8627	0.7806	16711648	0.46
RA	16	1	0.9961	0.8414	613872	12.61
RA	16	4	0.8118	0.75	2391024	3.24
RA	16	8	0.7216	0.75	3724272	2.08
RA	8	1	1	1	153528	50.42
RA	8	4	0.7373	0.75	613496	12.62
RA	8	8	0.8275	0.75	1229048	6.3
RA	4	1	1	1	38388	201.66
RA	4	4	1	1	153468	50.44
RA	4	8	0.7294	0.75	137980	56.1

Table 8: Results for GD Optimizer on the COSET dataset in section 2.1.2. k is the sub-kernel sample size, and s is the number of samples. The F1 and ROC scores are computed on the full, optimised kernel. Results are ordered by descending k and grouped by ansatze. Full kernels ($k = m$) are highlighted in bold.

Ansatz	k	s	ROC AUC	F1	Queries	Speed-up
HE	**96**	**1**	**1**	**1**	**5796768**	**1**
HE	32	1	1	1	1564640	3.7
HE	32	4	1	1	6262752	0.93
HE	32	8	1	1	12525536	0.46
HE	16	1	1	1	403184	14.38
HE	16	4	1	1	1842160	3.15
HE	16	8	1	1	3684336	1.57
HE	8	1	0.4902	0.5795	4408	1315.06
HE	8	4	1	1	425976	13.61
HE	8	8	0.7451	0.685	921080	6.29
HE	4	1	0.5804	0.5795	6036	960.37
HE	4	4	1	0.9688	75004	77.29
HE	4	8	1	1	104700	55.37
RA	**96**	**1**	**1**	**1**	**5796768**	**1**
RA	32	1	1	1	774624	7.48
RA	32	4	1	1	6262752	0.93
RA	32	8	1	1	12525536	0.46
RA	16	1	0.9529	0.875	311664	18.6
RA	16	4	1	1	1842160	3.15
RA	16	8	1	1	3684336	1.57
RA	8	1	0.7843	0.6161	28792	201.33
RA	8	4	0.9647	0.906	118136	49.07
RA	8	8	0.7569	0.75	921336	6.29
RA	4	1	1	0.874	22220	260.88
RA	4	4	1	0.906	107164	54.09
RA	4	8	0.702	0.6161	198268	29.24

Table 9: Reference Values for AF vs D. The best Classical Kernel Represents the result of grid search as described in Appendix A. Classical NN Reference is the result from [27]

	ROC	F1
Classical NN Reference	0.818	0.777
Best Classical Kernel	0.802	0.77
Quantum Fidelity Kernel	0.764	0.741

Table 10: Results for AF vs D dataset in section 2.1.3. Grouped by optimiser (ADAM, GD and SPSA) k is the sub-kernel sample size, and s is the number of samples. The F1 and ROC scores are computed on the full, optimised kernel. Results are ordered by descending k and grouped by ansatze. Full kernels ($k = m$) are highlighted in bold.

Optimizer	Ansatz	k	s	ROC AUC	F1	Queries	Speed-up
SPSA	**HE**	**1566**	**1**	**0.793**	**0.778**	**2.01 x 10^8**	**1**
SPSA	HE	16	1	0.786	0.778	8.10 x 10^5	248.02
SPSA	HE	16	4	0.793	0.778	0.81 x 10^6	61.61
SPSA	HE	8	1	0.774	0.774	4.06 x 10^5	494.25
SPSA	HE	8	4	0.788	0.781	0.41 x 10^6	127.57
SPSA	**RA**	**1566**	**1**	**0.745**	**0.774**	**1.85 x 10^8**	**1**
SPSA	RA	16	1	0.779	0.778	6.37 x 10^5	291.11
SPSA	RA	16	4	0.789	0.763	0.64 x 10^6	67.45
SPSA	RA	8	1	0.775	0.774	3.18 x 10^5	583.09
SPSA	RA	8	4	0.773	0.763	0.32 x 10^6	138
ADAM	**HE**	**1566**	**1**	**0.791**	**0.782**	**2.02 x 10^9**	**1**
ADAM	HE	16	1	0.777	0.774	1.1 x 10^7	182.47
ADAM	HE	16	4	0.787	0.774	1.1 x 10^7	45.65
ADAM	HE	8	1	0.789	0.782	5.58 x 10^6	361.05
ADAM	HE	8	4	0.784	0.767	0.56 x 10^7	89.30
ADAM	**RA**	**1566**	**1**	**0.774**	**0.778**	**1.30 x 10^9**	**1**
ADAM	RA	16	1	0.791	0.775	4.85 x 10^6	268.40
ADAM	RA	16	4	0.775	0.774	0.48 x 10^7	62.98
ADAM	RA	8	1	0.774	0.778	2.24 x 10^6	580.27
ADAM	RA	8	4	0.770	0.762	2.24 x 10^6	146.58
GD	**HE**	**1566**	**1**	**0.775**	**0.778**	**2.02 x 10^9**	**1**
GD	HE	16	1	0.789	0.776	1.13 x 10^7	178.41
GD	HE	16	4	0.782	0.782	1.13 x 10^7	42.28
GD	HE	8	1	0.751	0.782	5.73 x 10^6	351.83
GD	HE	8	4	0.775	0.782	0.57 x 10^7	88.20
GD	**RA**	**1566**	**1**	**0.783**	**0.778**	**3.75 x 10^8**	**1**
GD	RA	16	1	0.791	0.782	4.68 x 10^6	80.06
GD	RA	16	4	0.783	0.77	0.47 x 10^7	19.76
GD	RA	8	1	0.767	0.772	2.51 x 10^6	149.21
GD	RA	8	4	0.795	0.778	2.51 x 10^6	37.56

Table 11: Reference Values for NBU vs AFD. The best Classical Kernel Represents result of grid search as described in Appendix A. Classical NN Reference is the result from [27]

	ROC	F1
Classical NN Reference	0.868	0.828
Best Classical Kernel	0.859	0.809
Quantum Fidelity Kernel	0.850	0.807

Table 12: Results for NBU vs AFD dataset in section 2.1.3. Grouped by optimiser (ADAM, GD and SPSA) k is the sub-kernel sample size, and s is the number of samples. The F1 and ROC scores are computed on the full, optimised kernel. Results are ordered by descending k and grouped by ansatze. Full kernels ($k = m$) are highlighted in bold.

Optimizer	Ansatz	k	s	ROC AUC	F1	Queries	Speed-up
SPSA	**HE**	**2796**	**1**	**0.844**	**0.822**	**7.93 x 10^8**	**1**
SPSA	HE	16	1	0.834	0.826	1.13 x 10^6	702.02
SPSA	HE	16	4	0.843	0.828	4.03 x 10^6	174.29
SPSA	HE	8	1	0.833	0.822	5.74 x 10^5	1382.46
SPSA	HE	8	4	0.836	0.828	0.57 x 10^6	350.59
SPSA	**RA**	**2796**	**1**	**0.83**	**0.820**	**7.14 x 10^8**	**1**
SPSA	RA	16	1	0.837	0.819	9.03 x 10^5	790.26
SPSA	RA	16	4	0.833	0.82	0.90 x 10^6	194.75
SPSA	RA	8	1	0.841	0.822	4.48 x 10^5	1593.03
SPSA	RA	8	4	0.831	0.821	0.45 x 10^6	370.98
ADAM	**HE**	**2796**	**1**	**0.834**	**0.826**	**2.82 x 10^9**	**1**
ADAM	HE	16	1	0.833	0.824	1.54 x 10^7	182.64
ADAM	HE	16	4	0.85	0.823	1.54 x 10^7	43.07
ADAM	HE	8	1	0.826	0.824	7.71 x 10^6	365.04
ADAM	HE	8	4	0.840	0.824	0.77 x 10^7	93.90
ADAM	**RA**	**2796**	**1**	**0.83**	**0.824**	**2.82 x 10^9**	**1**
ADAM	RA	16	1	0.841	0.821	6.27 x 10^6	449.13
ADAM	RA	16	4	0.833	0.819	0.63 x 10^7	111.20
ADAM	RA	8	1	0.848	0.821	3.14 x 10^6	895.85
ADAM	RA	8	4	0.830	0.822	0.31 x 10^7	224.12
GD	**HE**	**2796**	**1**	**0.822**	**0.823**	**2.82 x 10^9**	**1**
GD	HE	16	1	0.836	0.826	1.58 x 10^7	178.66
GD	HE	16	4	0.837	0.828	1.58 x 10^7	42.07
GD	HE	8	1	0.835	0.828	7.92 x 10^6	355.37
GD	HE	8	4	0.839	0.821	0.79 x 10^7	90.22
GD	**RA**	**2796**	**1**	**0.833**	**0.822**	**2.82 x 10^9**	**1**
GD	RA	16	1	0.84	0.822	6.49 x 10^6	433.50
GD	RA	16	4	0.835	0.819	0.65 x 10^7	111.94
GD	RA	8	1	0.843	0.824	3.26 x 10^6	862.77
GD	RA	8	4	0.834	0.836	0.33 x 10^7	207.59