Supporting Information

Exploring Quantum Active Learning for Material Design and Discovery

Maicon Pierre Lourenço[1*], Hadi Zadeh-Haghighi[2], Jiří Hostaš[3,4], Mosayeb Naseri[3,4], Daya Gaur[5], Christoph Simon[2] and Dennis R. Salahub[4]

1- Departamento de Química e Física – Centro de Ciências Exatas, Naturais e da Saúde – CCENS – Universidade Federal do Espírito Santo, 29500-000, Alegre, Espírito Santo, Brasil.

2- Department of Physics and Astronomy, Institute for Quantum Science and Technology, Quantum Alberta, and Hotchkiss Brain Institute, University of Calgary, Calgary, AB T2N 1N4, Canada.

3- Digital Technologies Research Centre, National Research Council of Canada, 1200 Montréal Road, Ottawa, ON, K1A 0R6 Canada.

4- Department of Chemistry, Department of Physics and Astronomy, CMS Centre for Molecular Simulation, IQST Institute for Quantum Science and Technology, Quantum Alberta, QHA Quantum Horizons Alberta, University of Calgary, 2500 University Drive NW, Calgary, AB, T2N 1N4, Canada.

5- Department of Computer Science, University of Lethbridge, 4401 University Dr. West Lethbridge, AB T1K 3M4, Canada.

** Address correspondence to:* maiconpl01@gmail.com(MPL).

1. Statistical regression

The idea behind statistical regression is to obtain N *observed* properties **y** = (y$^{(1)}$, ...,y$^{(n)}$), i = 1, ..., N; to describe **y** statistically, the descriptor **x**$^{(i)}$ = (x$_1^{(i)}$, ...,x$_k^{(i)}$), with K variables is required. The property in our case is the piezoelectric coefficient and energy storage density for $Ba_{(1-x)}A_xTi_{(1-y)}B_yO_3$ perovskites and band-gap for $AA'BB'O_6$ perovskites. This results in a matrix **X** of dimension (N × K), called the feature matrix which is associated with the one-dimensional vector **y** (the objective function) of dimension (N). Here we define descriptor **x**$^{(j)}$ = (x$_1^{(j)}$, ...,x$_k^{(j)}$) in the virtual space ($N_{virtual}^{k}$): j = 1, ..., $N_{virtual}^{k}$; where *j* stands for the *j-th* virtual (non-observed) structure.

To model the desired problem in this manner, several surrogate models, such as support vector regression (SVR) or Gaussian Process regression (GP) can be used by utilizing high level libraries, such as scikit-learn[1].

After performing the regression, the statistical model is obtained and represented as:

$$\hat{y} = \hat{f}(\boldsymbol{X}), \qquad \textbf{\textit{(1)}}$$

where $\hat{y}$ is the vector with the predicted properties and $\hat{f}(\boldsymbol{X})$ is the statistical model (the predictor)designed from the SVR our GPR models.

Usually, to obtain a model without data bias, the matrices **X** and **y** – which define the initial data to obtain and test the ML models ($\mathbf{X}^{\mathrm{I}}$,$\mathbf{y}^{\mathrm{I}}$) – are split into two other matrices: ($\mathbf{X}^{\mathrm{train}}$, $\mathbf{y}^{\mathrm{train}}$), which are used to train the statistical model and ($\mathbf{X}^{\mathrm{test}}$, $\mathbf{y}^{\mathrm{test}}$) to validate it.

In order to obtain the average ($\mu(\mathbf{x}^{(i)})$ for the computed structures and $\mu(\mathbf{x}^{(j)})$ for the non-computed, virtual, structures) and the uncertainty (the standard deviation $\sigma(\mathbf{x}^{(i)})$ and $\sigma(\mathbf{x}^{(j)})$), the matrices ($\mathbf{X}^{\mathrm{train}}$, $\mathbf{y}^{\mathrm{train}}$) are partitioned K times for K-fold cross-validation (CV) or B times for a non-parametric bootstrap (BS). For each partition p (in a total space of P=K or P=B) a statistical model is obtained. Hence: $\hat{y}^{p} = \hat{f}(\boldsymbol{X})$, p = 1, ..., P. Then, for each descriptor in the observed data set **x**$^{(i)}$ (or for each descriptor j in the non-computed or virtual structures space: $\mathbf{x}^{(j)}$) the average $\mu(\mathbf{x}^{(i)})$ (or $\mu(\mathbf{x}^{(j)})$ for the virtual structures) and the standard deviation $\sigma(\mathbf{x}^{(i)})$ (or $\sigma(\mathbf{x}^{(j)})$) are obtained, as illustrated in figure S1 for the data in the observed space.

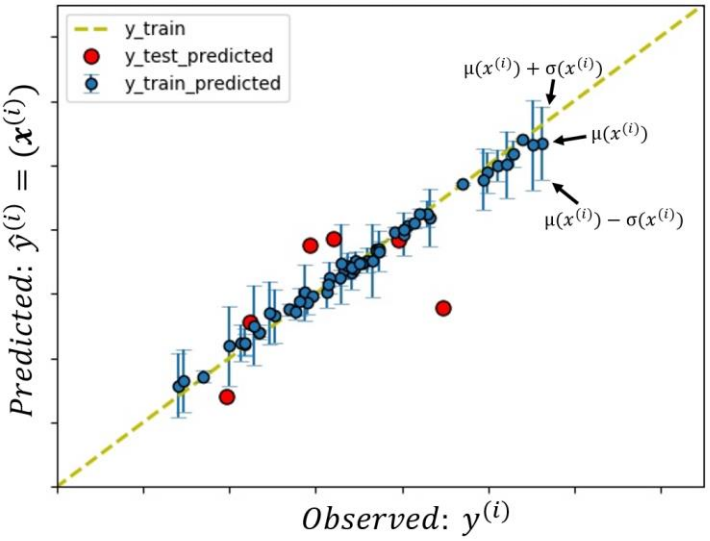

Figure S1- Illustrative plot of the observed $y^{(i)}$ and predicted $\hat{y}(x^{(i)})$ target property. The use of SVR allows us to have a regression model for each data point $x^{(i)}$ represented from the mean $\mu(x^{(i)})$ and the standard deviation $\sigma(x^{(i)})$. The abscissa is the observed property and the ordinate the predicted one. The same for the unexplored space descriptors for inference and decision making: exploitation ($\mu(x^{i})$) and exploration ($\sigma(x^{(j)})$).

The mean $\mu(\mathbf{x}^{(j)})$ and the standard deviation $\sigma(\mathbf{x}^{(j)})$ for each descriptor entry in the non-observed (virtual) space (whose dimension is defined by $N^{k}_{virtual}$) will be used to obtain the acquisition function[2, 3] which is used to indicate the next candidate to be computed. The next candidate is, then, incorporated in the initial descriptor matrix: ($\mathbf{X}^{\text{train+1}}$, $\mathbf{y}^{\text{train+1}}$) and the iteration process continues one step more until the optimization of the target property.

2. Quantum circuits used for data encoding

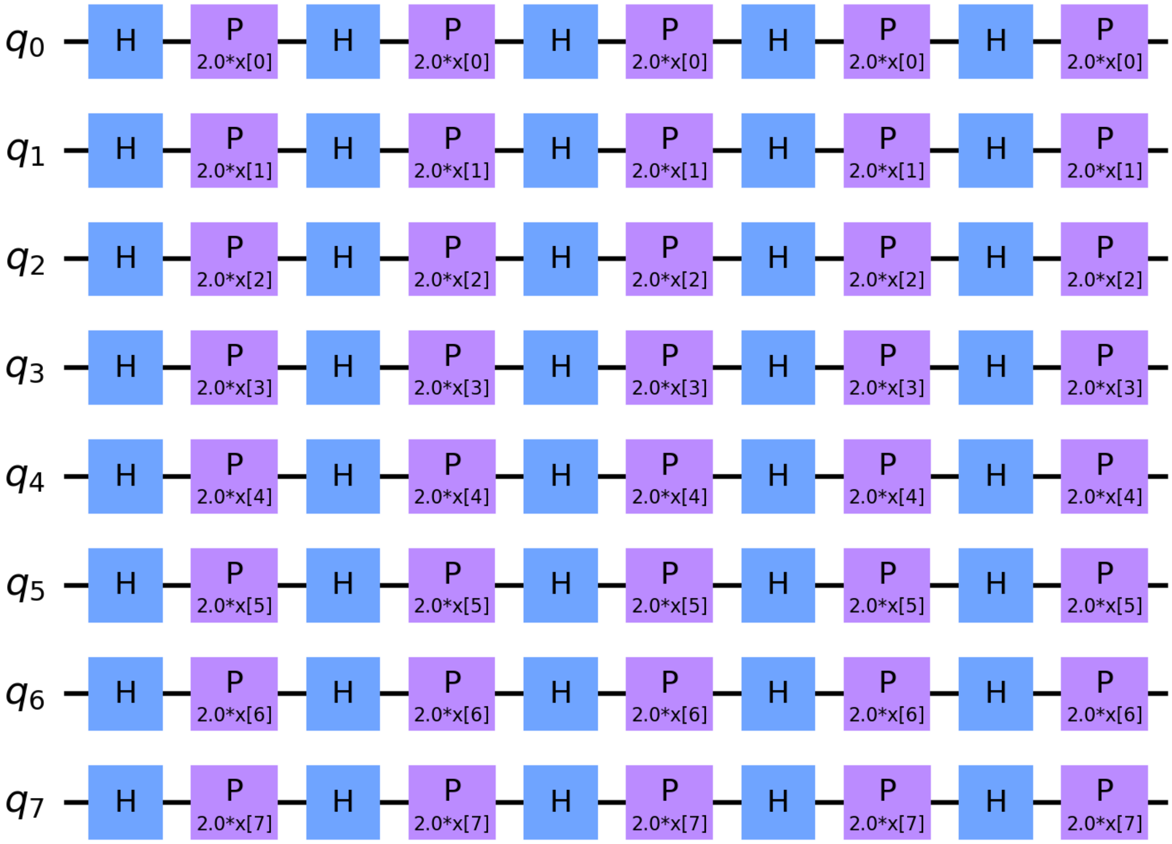

Figure S2- ZFeatureMap quantum circuit (QC) encoding as implemented in Qiskit framework. This QC was built with 8 feature maps or qubits (q0, q1, …, q7); the number of repeated circuits (reps) is equal to 5; the number of QC parameters, θ, is 8: x[0], …, x[7].

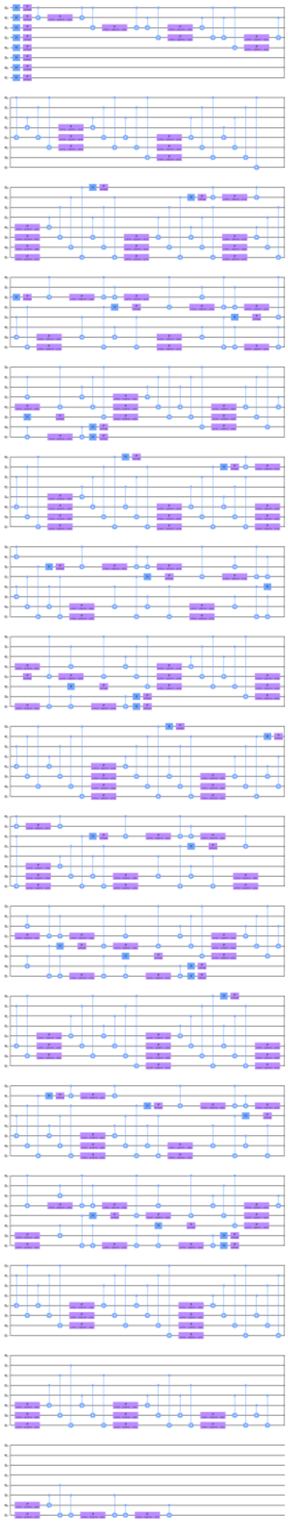

Figure S3- ZZFeatureMap quantum circuit (QC) encoding with full entanglement, as implemented in Qiskit framework. This QC was built with 8 feature maps or qubits (q0, q1, …, q7); the number of repeated circuits (reps) is equal to 5; the number of QC parameters, θ, is 8: x[0], …, x[7].

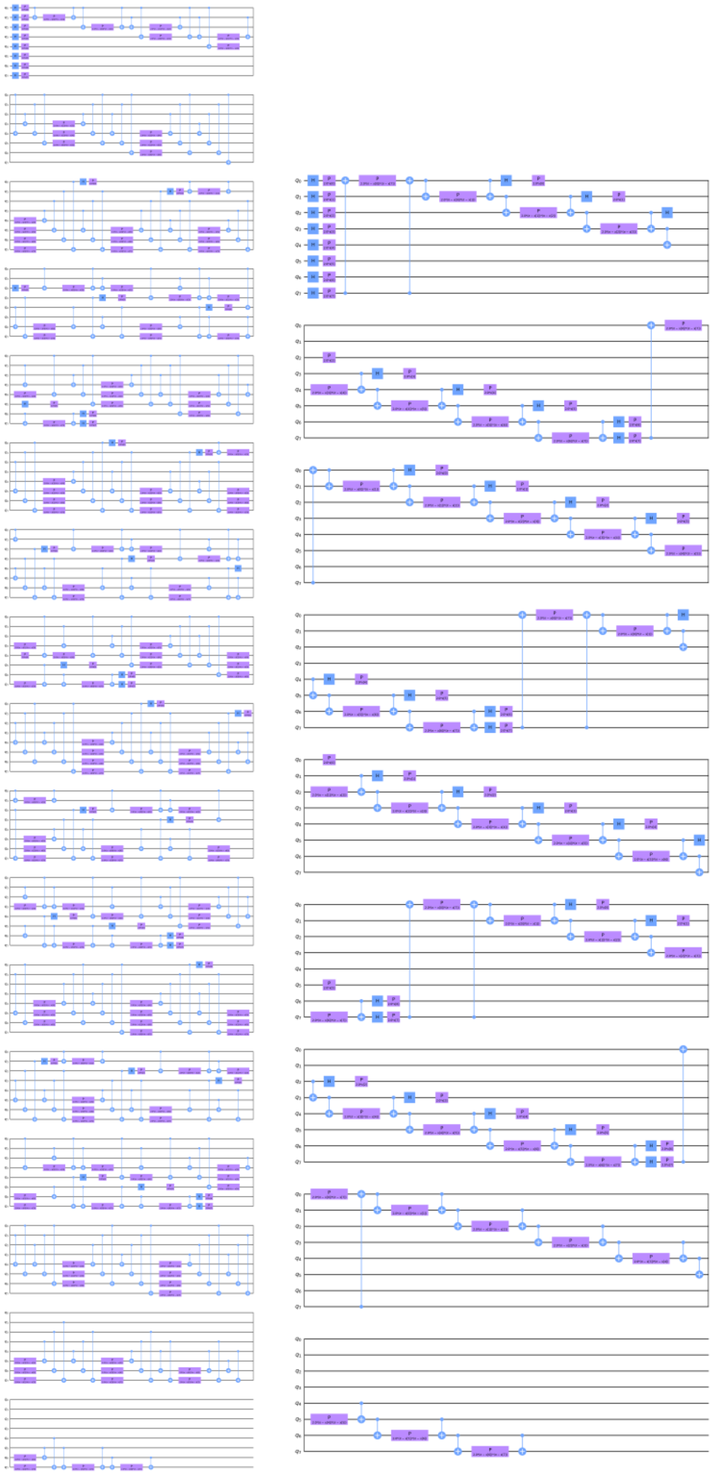

Figure S4- PauliFeatureMap quantum circuit (QC) encoding with full entanglement on the left and circular entanglement on the right, as implemented in Qiskit framework. This QC was built with 8 feature maps or qubits (q0, q1, …, q7); the number of repeated circuits (reps) is equal to 5; the number of QC parameters, θ, is 8: x[0], …, x[7].

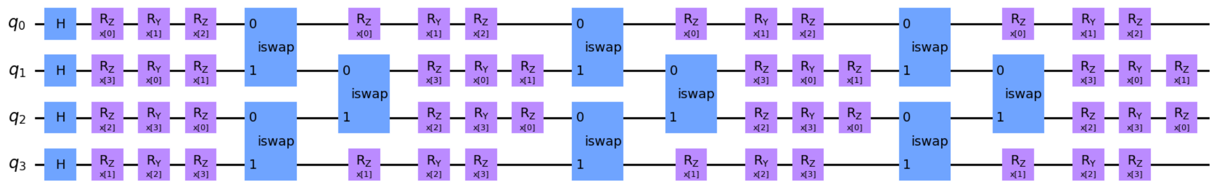

Figure S5- HighDim quantum circuit (QC) encoding as defined in sQUlearn framework. This QC was built with 4 feature maps or qubits (q0, q1, …, q3); the number of repeated circuits (reps) is equal to 4.

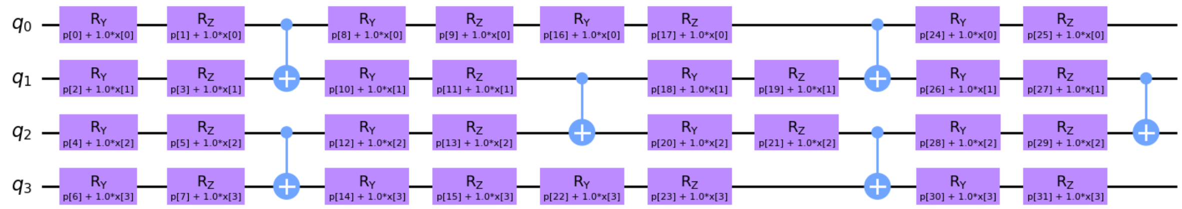

Figure S6- YZ_CX quantum circuit (QC) encoding as defined in sQUlearn framework. This QC was built with 4 feature maps or qubits (q0, q1, …, q3); the number of repeated circuits (reps) is equal to 4.

3. Grid search hyperparameterization for classical (SVR) and quantum (QSVR) machine learning models

The hyperparameters for SVR (using radial basis function kernel) and QSVR (using ZFeatureMap encoding and fidelity quantum kernel) were obtained using a gridsearch. After that, the best hyperparametes were selected and used in AL and QAL subsequent studies. This grid search is in line with the supporting information of Ref[4].

Piezoelectric coefficient of $Ba_{(1-x)}A_xTi_{(1-y)}B_yO_3$ perovskites (system I):

Data size: 22.

C and gamma hyperparameters grid for classical SVR:

'svr__C': [1.0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 200, ***1000***]
'svr__gamma': [0.001, 0.007, 0.008, 0.009, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08,

0.09, ***0.1***, 1, 10, 100]

C hyperparameter grid for QSVR using ZFeatureMap and fidelity quantum kernel:

'qsvr__C': [100, 200, 300, 400, 500, 600, 700, 800, 900, ***1000***, 1500, 2000]

The optimum hyperparameters are in bold.

Band-gap of $AA'BB'O_6$ double perovskites (system II):

Data size: 15.

C and gamma hyperparameters grid for classical SVR:

'svr__C': [***1.0***, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 200, 1000]
'svr__gamma': [***0.001***, 0.007, 0.008, 0.009, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 1, 10, 100]

C hyperparameter grid for QSVR using ZFeatureMap and fidelity quantum kernel:

'qsvr__C': [100, 200, 300, 400, 500, 600, 700, 800, 900, ***1000***]

The optimum hyperparameters are in bold.

Energy storage density of $Ba_{(1-x)}A_xTi_{(1-y)}B_yO_3$ perovskites (system III):

Data size: 73.

C and gamma hyperparameters grid for classical SVR:

'svr__C': [0.1, 1.0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, ***140***, 150, 200, 1000]
'svr__gamma': [10.0**(-4), 0.001, 0.007, 0.008, 0.009, 0.01, 0.02, 0.03, 0.04, 0.05, ***0.06***, 0.07, 0.08, 0.09, 0.1, 1, 10, 100]

C hyperparameter grid for QSVR using ZFeatureMap and fidelity quantum kernel:

'qsvr__C': [100, 200, 300, 400, 500, 600, 700, 800, 900, ***1000***]

The optimum hyperparameters are in bold.

4. Grid search hyperparameterization for classical (GPR) and quantum (QGPR) machine learning models

The hyperparameters for GPR (using "DotProduct + white" kernel) and QGPR (using HighDim encoding and projected quantum kernel) were obtained using a gridsearch. MBTR and spin multiplicity descriptors were used. Dimensionality reduction was then performed using PCA and the first 4 principal components were used as input features (descriptors) for both the GPR and the QGPR algorithms. Subsequently, the optimal hyperparameters were selected and used in the AL and the QAL studies. This grid search is in line with the supporting information of Ref[4].

Structural and spin search determination of 3Al@Si$_{11}$ [2, 4, 6]:

Data size: 60.

alpha hyperparameters grid for classical GPR:

'gpr__alpha': [1e15, 1e10, 1e5, 1e3, **10,** 1, 1e-1, 1e-3, 1e-5, 1e-10, 1e-15]

sigma hyperparameters grid for quantum GPRR:

'qgpr__sigma': [1e-6, 1e-5,1e-4, **1e-3**, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3]

The optimum hyperparameters are in bold.

5. Quantum machine learning set up

The regression mean (μ) and its epistemic uncertainty (σ) used in QAL were obtained from QSVR models, where the μ and σ were obtained by CV with 5-fold resampling (K = 5): CV5. From those, the acquisition function EI (Eqs. 8 and 9 in the main text) was evaluated to indicate the next perovskites to have their properties investigated or measured. The QSVR supervised learning regression algorithm (using different QC encoding to obtain the fidelity quantum kernel, FQK) is available in the current version of MLChem4D, which was developed in Python3.x and uses the scikit-learn library as well as Qiskit for quantum computing and QML. In the design loop, the ML regression fit is made on 95 % of the observed data (the training set) and tested on 5 % of it (the test set).

Tab. 1 presents the mean absolute error for the training and testing set of the systems (I), (II) and (III) obtained by different AL and QAL iterations. 5-fold cross validation was used. Also, the optimum hyperparameters are shown. They were obtained by grid search for a fixed perovskite database, as described in detail here in the SI.

Table 1. Statistics of the hyperparameters used in this work, as defined in the scikit-learn and Qiskit libraries using the ZFeatureMap for data encoding for the modified perovskites. 95 % of the data were randomly chosen for the training set and 5 % for testing for all materials studied, as defined during the QAL cycles. The cost function used to evaluate the quality of the regression was the mean absolute error (MAE) and it is presented for different data sizes obtained by QAL. "Hyper." means: hyperparameters used in the QSVR (using fidelity quantum kernel, FQK) or classical SVR (radial basis function, RBF, kernel) regression. The MAE for $Ba_{(1-x)}A_xTi_{(1-y)}B_yO_3$ piezoelectric coefficient (system I), band-gap of $AA'BB'O_3$ double-perovskites (system II) and energy storage density of $Ba_{(1-x)}A_xTi_{(1-y)}B_yO_3$ (system III) has the following units: pC/N, eV and mJ/cm^3, respectively. Quantum circuit features: "S" is superposition; "E" is entanglement; "NQ" is non-quantum (classic).

System I	**Hyper.**	**Quantum features**	**MAE train**	**MAE test**	**MAE train**	**MAE test**	**MAE train**	**MAE test**
			22 data		**~32 data**		**~42 data**	
$Ba_{(1-x)}A_xTi_{(1-y)}B_yO_3$	C = 10^3; ZFeatureMap; kernel = FQK	S	14.06	90.04	10.91	73.14	25.81	79.35
	C = 10^3; ZZFeatureMap; kernel = FQK	S + E	12.82	82.05	26.03	88.81	26.13	59.01

	C = 10^3; PauliFeatureMap; kernel = FQK	S + E	15.82	13.62	20.51	89.08	22.90	68.82
	C = 10^3; γ = 10^{-1}; kernel = RBF	NQ	16.25	87.11	25.65	59.16	24.84	80.91
System II	**Hyper.**		**MAE train**	**MAE test**	**MAE train**	**MAE test**	**MAE train**	**MAE test**
			10 data		**~20 data**		**~30 data**	
AA'BB'O_6	C = 10^3; ZFeatureMap; kernel = FQK	S	0.15	0.75	0.18	0.42	0.16	0.87
	C = 10^3; ZZFeatureMap; kernel = FQK	S + E	0.16	0.70	0.16	0.67	0.14	0.47
	C = 10^3; PauliFeatureMap; kernel = FQK	S + E	0.08	0.84	0.16	0.21	0.16	0.54
	C = 1; γ = 10^{-3}; kernel = RBF	NQ	0.16	0.11	0.22	0.49	0.21	0.36
System III	**Hyper.**		**MAE train**	**MAE test**	**MAE train**	**MAE test**	**MAE train**	**MAE test**
			73 data		**~93 data**		**~113 data**	
$Ba_{(1-x)}A_xTi_{(1-}$	C = 10^3; ZFeatureMap; kernel = FQK	S	2.06	6.31	1.93	4.68	1.93	9.95
	C = 10^3; PauliFeatureMap; kernel =	S + E	1.80	16.08	2.05	13.46	2.06	13.92

$_{y)}B_yO_3$	FQK							
	C = 140; γ = 0.06; kernel = RBF	NQ	4.77	7.30	4.40	5.32	4.43	5.86

The regression QSVR and SVR hyperparameters that resulted in the smallest mean absolute error (MAE) for the training and the testing set are shown in Tab. 1. The regularization parameter C for QSVR was obtained considering the ZFeatureMap data encoding for certain perovskites data (fixed) and was used in further QAL studies with ZZFeatureMap and PauliFeatureMap data encoding with entanglement – as shown in Tab. 1. For classical SVR (used in AL), the optimum hyperparameters presented in Tab. 1 – regularization parameter (C) and the kernel coefficient (γ) for the radial basis function kernel (RBF) – were optimized in a grid search (also for a fixed data) with tolerance factor equal to 10^{-3} and ε = 0.01. The optimum hyperparameters found were used subsequently in the AL (classical) study, as shown in Tab. 1. Further information can be found in the documentation of the scikit-learn library. Also, additional information about statistical regression, classical (RBF) and quantum (FQK) kernels can be found here in the SI.

Table 2. Statistics of the hyperparameters used in this work, as defined in the scikit-learn and sQUlearn libraries for the 3Al@Si$_{11}$ nanoparticle. 95 % of the data were randomly chosen for the training set and 5 % for testing for 3Al@Si$_{11}$, as defined during the QAL cycles using exploitation for decision making. The cost function used to evaluate the quality of the regression was the mean absolute error (MAE) and it is presented for different data sizes obtained by QAL. "Hyper." means: hyperparameters used in the QGPR (using projected quantum kernel, PQK, or fidelity quantum kernel, FQK) or classical GPR (radial basis function, RBF, kernel) regression. The MAE is in Hatree × 10^{-2}. The MBTR descriptor was used and PCA was applied, allowing a dimensionality reduction of 4. Quantum circuit features: "S" is superposition; "E" is entanglement; "NQ" is non-quantum (classic).

System	**Hyper.**	**Quantum features**	**MAE train**	**MAE test**	**MAE train**	**MAE test**	**MAE train**	**MAE test**

IV								
			20 data		**~100 data**		**~200 data**	
3Al@Si$_{11}$	featureMap = HighDim; Kernel = PQK; σ=0.001	S + E	0.12	7.90	0.28	3.36	0.43	3.48
	featureMap = YZ_CX; Kernel = PQK; σ=0.001	S + E	0.12	6.58	0.31	2.74	0.42	3.10
	Kernel = DotProduct + WhiteKernel, σ[1.0, (10^{-3}-10^{3})] , noise[10.0, (10^{-3}-10^{3})]; α = 10	NQ	1.15	0.07	2.96	3.62	3.06	4.00

The GPR and QGPR hyperparameters used during the QAL – as defined in scikit-learn and sQUlearn) that resulted in the smallest mean absolute error (MAE) for the training and the testing set are shown in Tab. 2.

6. *Single-perovskite descriptor*

The set of descriptors for non-stoichiometric perovskites – as for $Ba_{(1-x)}A_xTi_{(1-y)}B_yO_3$ for the energy storage density search – is composed of the weighted atomic properties ($\bar{P}_J$): $\bar{P}_J = \sum_J^{N_{ions}} x_J P_J$; where x_J is the fraction of ions that compose the sites A or B of the ABO_3 perovskite and P_J is the property of the Jth ion at the A or B site. N_{ions} is the number of ions in the A or B sites. Also, the following tolerance factors[5] are used as descriptor: $t_f = (\bar{r}_A + r_O)/[\sqrt{2}(\bar{r}_B + r_O)]$ and $t_t^{new} = \left(\frac{\bar{r}_O}{\bar{r}_B}\right) - Q_A\left[[Q_A - \left(\frac{\bar{r}_A}{\bar{r}_B}\right)/ln\left(\frac{\bar{r}_A}{\bar{r}_B}\right)\right]$; where Q_A is the oxidation state of site A and $\bar{r}_A$, and $\bar{r}_B$ are the average atomic radius of the A and B sites, respectively, obtained by $\bar{P}_J$; r_O, is the ionic radius of oxygen.

The set of descriptors developed in this work is composed of a list with the following elements:

(I) The tolerance factors t_f and t_t^{new}.

(II) The following properties of the A ($\bar{P}_A$) and B ($\bar{P}_A$) sites of ABO_3: (1) Shannon ionic radius[6, 7]; (2) ideal bond length of A-O and B-O; (3) electronegativity; (4) van der Waals radius; (5) first ionization energy; (6) molar volume; (7) atomic number; (8) atomic mass. As they are for both the A and B sites, 2×8 = 16 features result.

(III) The properties related to the A and B sites, as described in II, divided ($\bar{P}_A/\bar{P}_B$) and multiplied ($\bar{P}_A.\bar{P}_B$); resulting in another 16 features. Therefore, the set of descriptors developed is composed of a list with 34 properties (2 + 16 + 16).

The aforementioned atomic properties used to create sets of descriptors for the perovskites were obtained from the Python Materials Genomics[8] (pymatgen) library.

7. *Double-perovskite descriptor*

The descriptor for double-perovskites compounds is based on weighted atomic properties described in the previous section: SI-4. Since we need to include the information about both: the parent ABO_3 elements and the doped A' and B' sites, it results in effectively doubling the size of the descriptor when compared to the single perovskites. More specifically, 8 properties for each A, A', B, and B' element lead to a feature vector of length 32. Next, the properties were divided ($\bar{P}_A/\bar{P}_B, \bar{P}_{A'}/\bar{P}_{B'}$) and multiplied ($\bar{P}_A.\bar{P}_B, \bar{P}_{A'}.\bar{P}_{B'}$) effectively doubling the size (64) of the descriptor. More information about this descriptor can be found in our ICLR 2023 Workshop publication[9] and the dataset can be downloaded at its associated Github page (https://github.com/jiri-hostas/EDA-and-ML-for-Perovskites). The materials follow the $A_xA'_{(1-x)}B_yB'_{(1-y)}O_6$ formula, where:

A = [Ba, Bi, Ca, Dy, Eu, Gd, La, Li, Lu, Na, Pr, Sm, Sr, Tb, Y, Cu, Mg]

and

B = [Ag, Al, Bi, Ce, Co, Cr, Dy, Eu, Fe, Ga, Gd, In, La, Mg, Mn, Nd, Ni, Pr, Sc, Sm, Ta, Ti, Zn, I, Mo, Nb, Te, V, W]

8. Spin multiplicity and structural descriptors

The spin-multiplicity descriptor (SMD) is represented by the vector [(2S + 1), S, $2S\times(S+1)^{1/2}$, $n_{e\text{-unpaired}}$], where S is the total spin quantum number; (2S + 1) is the multiplicity; $2S\times(S+1)^{1/2}$ is the spin magnetic moment; $n_{e\text{-unpaired}}$ is the number of unpaired electrons. Take, for example, a cluster with SM equal to 4 (a quartet), we have: (2S + 1) = 4; S = 3/2; $2S\times(S+1)^{1/2} = 3.87$; $n_{e\text{-unpaired}} = 3$. Thus, the vector is: [4, 1.5, 3.87, 3]. The SMD descriptor is simply added to the structural-feature vector so that it can be used in supervised ML or QML where both structural and electronic structure descriptors are incorporated. Then, the resulting vector is transformed by principal component analysis (PCA), as implemented in scikit-learn. Thus, the spin multiplicities (SMs) are accounted for inside the AL or QAL loop for the structural determination.

The many-body tensor representation descriptor (MBTR) was used to represent the structures of the $3Al@Si_{11}$ nanoparticle isomers or homotops. The MBTR is implemented in the DScribe[10] library which is interfaced in QMLMaterial[11]. MBTR descriptor encodes information about the whole atomic structure and can be used to predict properties involving molecular energies. In the MBTR, a configuration of k atoms is transformed into a single scalar by using a geometry function g_k. Then kernel density estimations with a Gaussian kernel are employed to expand the scalar value.

9. Tutorial video explaining how to use the MLChem4D and QMLMaterial programs

YouTube video explaining how to run the classical Active Learning (AL) and Quantum Active Learning (QAL) for piezoelectric coefficient maximization of ABO_3 modified perovskites using SVR and QSVR: https://youtu.be/jLhFajQ0Uus. Example for running QMLMaterial with classical AL and QAL for $3Al@Si_{11}$ with spin multiplicities 2, 4 and 6 using GPR and QGPR: https://youtu.be/lVFI_XZxcyo.

10. References

1. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and É. Duchesnay, *J. Mach. Learn. Res.*, 2011, **12**, 2825-2830.
2. T. Lookman, P. V. Balachandran, D. Xue and R. Yuan, *npj Computational Materials*, 2019, **5**, 21.
3. D. R. Jones, M. Schonlau and W. J. Welch, *Journal of Global Optimization*, 1998, **13**, 455-492.
4. H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven and J. R. McClean, *Nature Communications*, 2021, **12**, 2631.
5. C. J. Bartel, C. Sutton, B. R. Goldsmith, R. Ouyang, C. B. Musgrave, L. M. Ghiringhelli and M. Scheffler, *Science Advances*, 2019, **5**, eaav0693.
6. R. D. Shannon and C. T. Prewitt, *Acta Crystallographica Section B*, 1969, **25**, 925-946.
7. R. Shannon, *Acta Crystallographica Section A*, 1976, **32**, 751-767.
8. S. P. Ong, W. D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V. L. Chevrier, K. A. Persson and G. Ceder, *Computational Materials Science*, 2013, **68**, 314-319.
9. J. Hostas, M. P. Lourenco, J. Garcia, H. Shahmohamadi, A. Tchagang, K. Shankar, V. Thangadurai and D. R. Salahub, presented in part at the Workshop on "Machine Learning for Materials" ICLR 20232023.
10. L. Himanen, M. O. J. Jäger, E. V. Morooka, F. Federici Canova, Y. S. Ranawat, D. Z. Gao, P. Rinke and A. S. Foster, *Comput Phys Commun*, 2020, **247**.
11. M. P. Lourenço, L. B. Herrera, J. Hostaš, P. Calaminici, A. M. Köster, A. Tchagang and D. R. Salahub, *Journal of Chemical Theory and Computation*, 2023, **19**, 5999-6010.