Quantum circuit learning as a potential algorithm to predict experimental chemical properties

Kan Hatakeyama-Sato, [a]* *Yasuhiko Igarashi,* [b] *Takahiro Kashikawa,* [c] *Koichi Kimura,* [c] *Kenichi Oyaizu* [a]*

[a]Department of Applied Chemistry, Waseda University.Tokyo 169-8555, Japan

[b]Faculty of Engineering, Information and Systems, University of Tsukuba, 1-1-1 Tennodai, Tsukuba 305-8573, Japan

[c]Fujitsu Ltd., Kanagawa 211-8588, Japan

*K.H.: satokan@toki.waseda.jp,

*K.O.: oyaizu@waseda.jp

KEYWORDS

Quantum computing, machine learning, materials informatics

ABSTRACT

We introduce quantum circuit learning (QCL) as an emerging regression algorithm for chemo- and materials-informatics. The supervised model, functioning on the rule of quantum mechanics, can process linear and smooth non-linear functions from small datasets (< 100 records). Compared with conventional algorithms, such as random forest, support vector machine, and linear regressions, the QCL can offer better predictions with some one-dimensional functions and experimental chemical databases. QCL will potentially help the virtual exploration of new molecules and materials more efficiently through its superior prediction performances.

Introduction

Recent data-oriented science provides chemical and material research insights by treating versatile experimental data with statistical tools.[1-5] Compared to human-based data recognition and accumulation, experimental data are analyzed and stored more objectively by data science.[1, 3] Such a data-oriented approach offers a more solid platform for interdisciplinary research conducted by chemists, data scientists, physicists, and experts in any other fields.[3, 5]

Evaluating molecular and material properties is essential in materials and chemo-informatics.[4, 5] Various supervised models, trained to predict specific properties from explanatory variables by learning their statistical relationships, have been developed in machine learning.[5] There are many supervised models represented by linear algorithms, support vector machines, and decision tree-based ensembles.[5] Also, recent deep learning technology of neural networks has broken the limit of prediction accuracy by drastically increasing the model complexity.[6] In chemical and material fields, appropriate use of such supervised models afforded the prediction of versatile material and chemical properties, such as conductivity,[7, 8] energy level,[9] photoconversion efficiency,[10] and toxicity.[4] Their prediction accuracy can exceed human experts and traditional computational simulations, gradually forming a solid platform for data-oriented science.[4, 5, 8-10]

On the other hand, most data science projects still have difficulty with the reliable prediction of experimental properties. The main challenges are a) the lack of trainable records and b) complex molecular interactions. Due to the high cost of actual experiments, typical database sizes of material projects are around $10^1 - 10^2$,[7] whereas deep learning mainly targets databases with over 10^4 records.[4-6] Although recent deep learning approaches, represented by fine-tuning (transfer

learning) and multipurpose learning, may offer an opportunity for small datasets, they may not be the complete solutions due to the still insufficient and diverse material databases to learn.[2, 7, 11]

As a promising approach, sparse modeling aims to extract linearity between explanatory and target variables.[12, 13] The method is powerful against small datasets owing to the simple linear assumption, and successful data analyses have been reported.[12, 14] Still, material and chemical properties appear from complex, non-linear atomic interactions.[5, 8] The linear approximation would face difficulties in expressing the non-linear system.

Here, we introduce quantum circuit learning (QCL), an emerging algorithm for supervised learning.[15-18] QCL works on the rule of quantum mechanics. It can predict various parameters likewise to classical models. The current quantum systems (noisy intermediate-scale quantum computers: NISQ)[19, 20] face the problems of calculation noise and the limited processable number of information units (qubits). Nevertheless, the advantage of quantum nature would appear under the support of classical computers.[19, 20]

Quantum machines and simulators have been examined in several fields, including quantum chemistry, combinatorial optimization, and machine learning.[21-26] Quantum neural networks, such as autoencoders, and generative adversarial networks, are the main prototypes of quantum machine learning.[21-24] They offer new potentials as supervised or unsupervised algorithms.

Since QCL is a frontier for machine learning, few reports have been reported on authentic regression tasks.[15, 17, 27] The success of prediction with a toxicity dataset of organic molecules was reported,[17] whereas the study was still conceptual. The prediction processes and advantages have been unclear, especially from chemical and material viewpoints.

Here, we conducted a more comprehensive study on QCL with standard datasets of one-dimensional functions and chemical properties. Both simulated and actual quantum computing was undertaken to clarify the challenges of QCL. The comparison with conventional models contrasted the benefits of the new approach: capable of learning both linear and non-linear functions even from small datasets. The property was also favorable for predicting the so-called extrapolating data region, essential for chemical and material research.

1. Theory of quantum circuit learning

First, we briefly explain the basic idea of quantum computing and circuit learning,[28] especially for readers unfamiliar with quantum systems. Quantum computers have qubits to maintain the information, whereas standard computers process information with bits (0 or 1). A qubit does not have one fixed state but supports multiple states probabilistically (superposition properties). Mathematically, the condition can be described by a two-dimensional complex vector.

In the case of an n-qubit system, a 2^n-dimensional complex vector is needed for the qubit state expression.[28] This seems confusing from the viewpoint of standard Euclidean space, but is required to express complex interactions of qubits, called quantum entanglement. Quantum systems become more complicated in exponential speed along with n, affording massively parallel computing, which is harder to be simulated by classical computers.

In a similar way to conventional machine learning, QCL treats a task of $\hat{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x})$,[15] where $\hat{y}$ is the predicted value for a target parameter y, $\boldsymbol{x}$ is explanatory variable, and $\boldsymbol{\theta}$ is trainable parameter. Generation of $f_{\boldsymbol{\theta}}$ only by a current quantum system (NISQ) is not feasible due to the limited computation power. Currently, only the prediction part of $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ is assigned to the quantum system

(or simulator), and other parts, such as loss calculation (e.g., $(\hat{y}-y)^2$) and parameter optimization, are done by classical computers.[15]

A mathematical expression of QCL for regression is not so complex (Figure 1, see Experimental section for derivation). The initial 2^n-dimensional quantum state vector, expressed by $(1,0,\ldots,0)^T$, is transformed to another state $\boldsymbol{w}=(w_1,w_2,\ldots,w_{2^n})^T$ by multiplying two complex operational matrices of $V(\boldsymbol{x})$ and $U(\boldsymbol{\theta})$ (**Eq 1**). Then, $\hat{y}$ is calculated from the squares of $w_1,w_2,\ldots,w_{2^n}$ (**Eq 2**).

	$$\boldsymbol{w}=\begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{2^n} \end{pmatrix}=U(\boldsymbol{\theta})V(\boldsymbol{x})\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$ $$=\begin{pmatrix} u_{1,1}(\boldsymbol{\theta}) & \cdots & u_{1,n^2}(\boldsymbol{\theta}) \\ \vdots & \ddots & \vdots \\ u_{n,1}(\boldsymbol{\theta}) & \cdots & u_{n^2,n^2}(\boldsymbol{\theta}) \end{pmatrix}\begin{pmatrix} v_{1,1}(\boldsymbol{x}) & \cdots & v_{1,n^2}(\boldsymbol{x}) \\ \vdots & \ddots & \vdots \\ v_{n,1}(\boldsymbol{x}) & \cdots & v_{n^2,n^2}(\boldsymbol{x}) \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$ $$\left(w_i,u_{i,j},v_{i,j}\in\mathbb{C}\right)$$	**Eq 1**
	$$\hat{y}=f_{\boldsymbol{\theta}}(\boldsymbol{x})=\sum_{i=1}^{2^{n-1}}|w_i|^2-\sum_{i=2^{n-1}+1}^{2^n}|w_i|^2$$	**Eq 2**

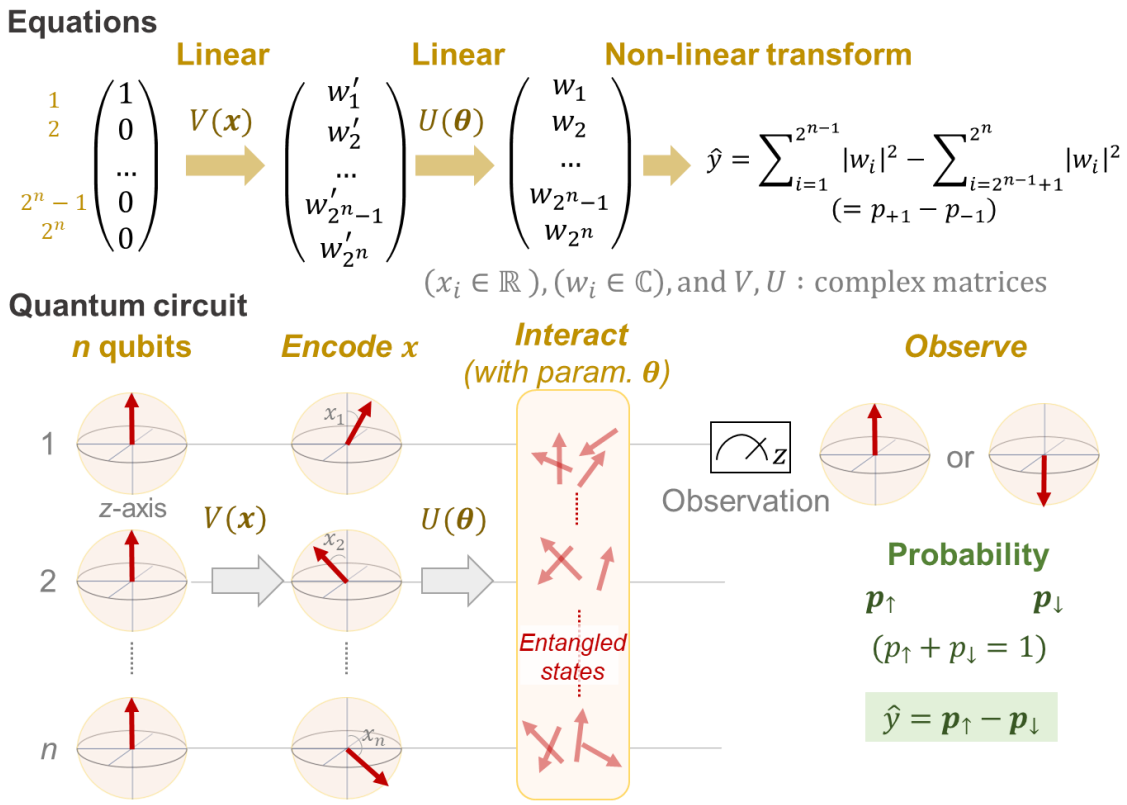


Figure 1 Mathematical and visual expression of a quantum circuit. Upward vectors in the initial state (mathematically, $(1,0,\ldots,0)^T$) are rotated by encoding gates of $V(\boldsymbol{x})$. In the figure, a simple case is shown where the i-th qubit is rotated along with the angle of x_i. Then, qubits interact via the gates of $U(\boldsymbol{\theta})$. Finally, up- or downward vector for the 1st qubit is observed with the probabilistic distribution. The transformations of $V(\boldsymbol{x})$ and $U(\boldsymbol{\theta})$ are linear against the initial state vector. The final observation is a non-linear step.

In a quantum circuit, $V(\boldsymbol{x})$ and $U(\boldsymbol{\theta})$ correspond to the encoding and interaction steps of qubits, respectively (Figure 1). Before calculation, all directions of qubit vectors are set upward along with the z-axis (corresponding to a vector of $(1,0,\ldots,0)^T$). The vectors are changed by the rotation gates of $V(\boldsymbol{x})$. The encoded states are further rotated and interacted with according to another matrix $U(\boldsymbol{\theta})$. Finally, the direction of one qubit (or theoretically any number of qubits) is observed to obtain the interacted result.[15, 17] The regression model can learn a variety of functions because

of the universality of quantum circuit[28] and the non-linear transformation steps during prediction (i.e., $\boldsymbol{x} \mapsto V(\boldsymbol{x})$, $\boldsymbol{\theta} \mapsto U(\boldsymbol{\theta})$ and final prediction by **Eq 2**).[15]

In **Eq 1**, naïve determination of V and U is not easy because they are $2^n \times 2^n$-dimensional matrices (i.e., over 10^{18} parameters with n = 30). In QCL, the matrices are prepared by repeated products of elementary gate components, such as $R_{i,x}(t)$, $R_{i,y}(t)$, and $\mathrm{CNOT}_{i,j}$ (**Eq 3**, Figure 2).

	$V(\boldsymbol{x}) = \prod A(x_k)$ $U(\boldsymbol{\theta}) = \prod A(\theta_k)$ $\left(A = R_{i,x}, R_{i,y}, \text{or } \mathrm{CNOT}_{i,j}\right)$	**Eq 3**

$R_{i,x}(t)$ and $R_{i,y}(t)$ are rotation gates, changing the i-th qubit state and affecting nothing against the others.[28] One qubit state (without entanglement) can be visualized as an arrow in a sphere (Bloch sphere, Figure 2a). The gates change the angle of an i-th qubit along with the x- (or y-) axis (**Eq 4**, **Eq 5**). A $\mathrm{CNOT}_{i,j}$ gate can switch the state of the j-th qubit according to the condition of the i-th qubit (Figure 2b, **Eq 6**). The gate is similar to an XOR operation in classical circuits. The three components are known as universal gate sets, which can make an arbitrary quantum circuit from their products.[28]

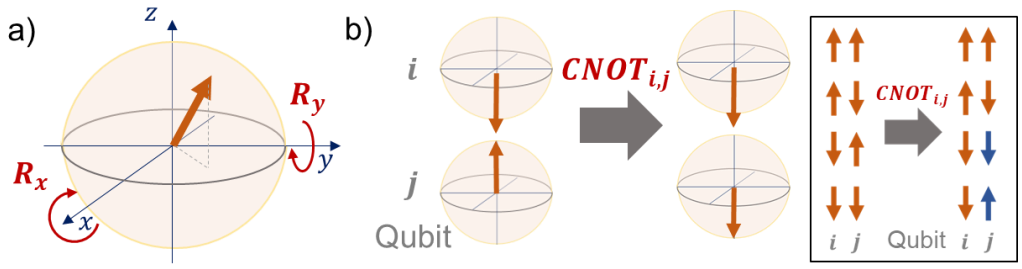


Figure 2 Visual understanding of quantum gates with Bloch spheres. a) R_x and R_y gates. A vector is rotated along with the *x*- or *y*-axis. b) CNOT gate. When the *i*-th qubit is upward, nothing happens to another qubit. When the *i*-th qubit is downward, the *j*-th qubit is flipped by the CNOT gate.

	$R_x(t) = \begin{pmatrix} \cos\left(\frac{t}{2}\right) & -i\sin\left(\frac{t}{2}\right) \\ -i\sin\left(\frac{t}{2}\right) & \cos\left(\frac{t}{2}\right) \end{pmatrix}$ $R_{i,x}(t) = I_2 \otimes I_2 \otimes \ldots \otimes R_x(t) \otimes I_2 \otimes \ldots \otimes I_2$ ($R_x(t)$ appears at the *i* th occurrence. I_2 is unit matrix)	**Eq 4**
	$R_y(t) = \begin{pmatrix} \cos\left(\frac{t}{2}\right) & -\sin\left(\frac{t}{2}\right) \\ \sin\left(\frac{t}{2}\right) & \cos\left(\frac{t}{2}\right) \end{pmatrix}$ $R_{i,y}(t) = I_2 \otimes I_2 \otimes \ldots \otimes R_y(t) \otimes I_2 \otimes \ldots \otimes I_2$	**Eq 5**
	$\mathrm{CNOT}_{i,j} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ (an example case of $i = 1$ and $j = 2$)	**Eq 6**

Due to the restriction of quantum physics, the interacted state $\boldsymbol{w}$ itself is not observable by actual quantum systems. Instead, other parameters, such as the probabilities of upward (↑) or downward (↓) eigenstates, are experimentally observable ($p_\uparrow$ and $p_\downarrow$, respectively, Figure 1). During actual quantum computing, such eigenstates are sampled via repeated calculations, and the probability difference between the two is calculated to obtain $\hat{y}$ (**Eq 7**).

	$\hat{y} = p_\uparrow - p_\downarrow$	**Eq 7**

	$$p_{+1} = \sum_{i=1}^{2^{n-1}} \lvert w_i \rvert^2$$ $$p_{-1} = \sum_{i=2^{n-1}+1}^{2^n} \lvert w_i \rvert^2$$	

For clearer understanding, we calculated analytical solutions of $\hat{y}$ with some simple quantum circuits (Table S1). An example quantum circuit, encoding $\boldsymbol{x} = (x_1, x_2)^T$ by four rotational gates, and interacting by one CNOT and one rotational gate, is shown in Figure 3a. Even the simple circuit gives a very complex analytical solution of $\hat{y}$ from **Eq 2**, consisting of repeated trigonometric functions (Figure 3a). The complexity should mathematically correspond to the superparallel and entangled nature of the quantum system.

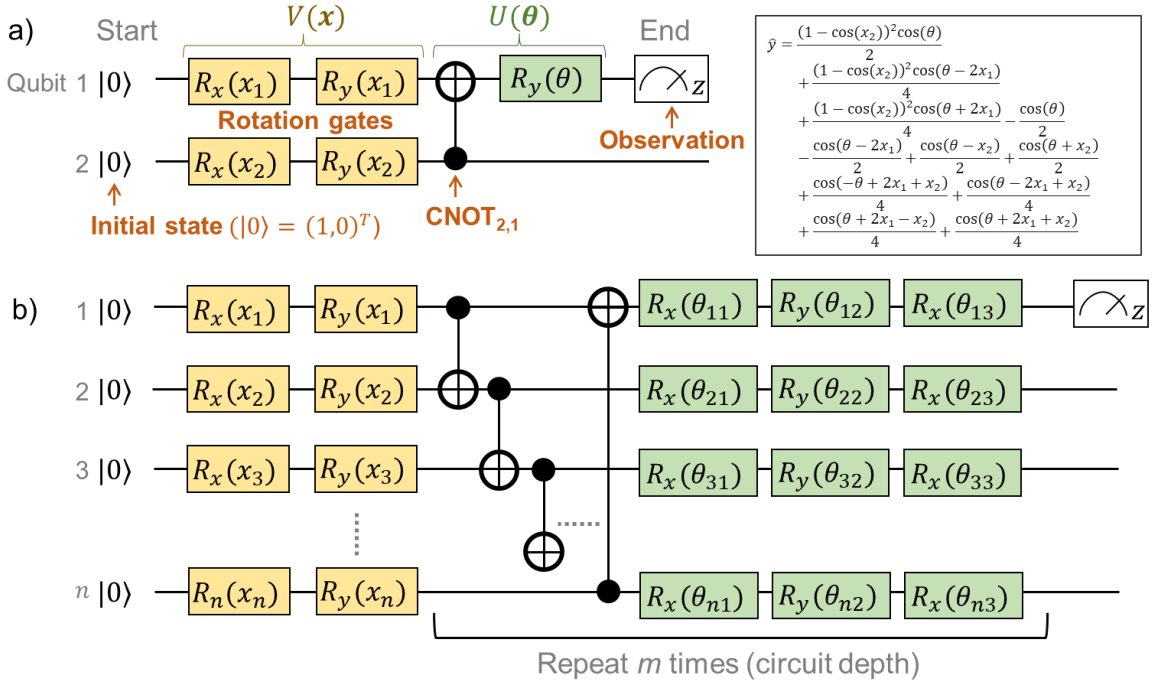


Figure 3 Quantum circuits for QCL. a) An example circuit of $V(\theta) = R_{1,y}(\theta) \cdot \mathrm{CNOT}_{2,1}$ and $V(x) = R_{1,y}(x_1) \cdot R_{1,x}(x_1) \cdot R_{2,y}(x_2) \cdot R_{2,x}(x_1)$. The analytical solution of $\hat{y}$ is also shown in the figure. b) General QCL circuit. The *i*-th qubit is encoded with $R_y(x_i)R_x(x_i)$. Then, repeated layers of CNOT and rotational gates are prepared for *m* times.

Regardless of the complex equation of $\hat{y}$ in QCL, the value always ranges from -1 to +1 because of the unitary nature of the operational matrices, V and U ($V^\dagger V = I, U^\dagger U = I$, where † indicates complex conjugates).[15, 28] This study tries to clarify the actual effects of such complex yet systematic prediction algorithms for various regression tasks.

2. One-dimensional regression

We examined the basic regression properties of QCL models with simple one-dimensional functions of $y = x,\ \sin(x)$, and e^{x-1} (Figure 4a). There are three major approaches to obtain $\hat{y}$: a) calculate state vectors (**Eq 1**, **Eq 2**) by classical computers, b) sample frequencies of upward and downward eigenvalues to get their probabilities ($p_\uparrow$ and $p_\downarrow$, **Eq 7**) by classical computers, and c) conduct the sampling by real quantum computers. Currently, many works obtain values by a) or b) due to the limited power of the quantum system.[15, 20-24] In the future, the role of c) will be more critical with larger qubits because of the exponential calculation cost (in the order of 2^n, **Eq 1**). Here, predictions were mainly made by the most precise method of a).

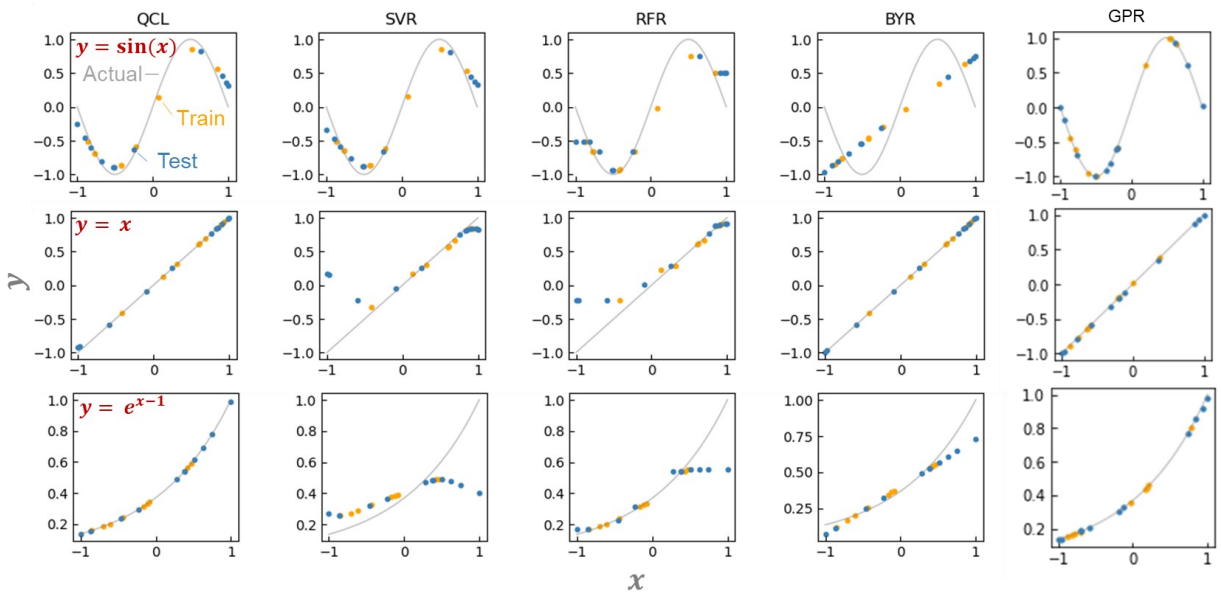


Figure 4 Predicting the function of $y = \sin(x)$, x, or, e^{x-1} using QCL, support vector machine for regression (SVR), random forest regression (RFR), Bayesian ridge regression (BYR), and Gaussian process regressor (GPR) (RBF + Dot-product). In the case of QCL, circuit parameters of $n = 2$ and $m = 3$ are chosen. Predictions were made from state-vector calculations. Other results are shown in Figure S 2. b) Predicting the function of $y = \sin(x)$ by an actual quantum computer (IBM Quantum) with $m = 2, 3,$ or 4. Models were trained by state-vector calculations. Full results are shown in Figure S 10.

Preliminary optimization of quantum circuits[15, 17] and our optimization revealed that the following configuration was concise and practical for regression tasks (for details, see results and explanations in Figure S 1). First, the inputted value $\boldsymbol{x}$ should be encoded by two rotational gates $R_{i,y}(x_i)$ and $R_{i,x}(x_i)$. Then, neighboring qubits had to be interacted CNOT gates, with $\boldsymbol{\theta}$ −dependent gate rotation $R_{i,y}(\theta_j)R_{i,x}(\theta_{j+1})R_{i,y}(\theta_{j+2})$. The CNOT interaction and rotation should be repeated *m* times (Figure 3b, Figure S 1).

Some notes on circuit design should be mentioned. Mitarai et al. proposed[15] the use of $\sin^{-1}$ or $\cos^{-1}$ to preconvert x_i for linear encoding of the inputted value (Table S1). On the other hand, we

noticed that the conversion induced unfavorable bending of $\hat{y}$ around$|x_i| \cong 1$, giving prediction errors (Figure S 1). The bending was caused by the large curvature of the inverse functions $(\sin^{-1}, \cos^{-1})$ near ± 1.

For qubit interaction, three gates, such as R_y, R_x, R_y, were introduced for one qubit. The selection was because at least three gates are needed for arbitrary rotation for the complex vectors (i.e., X-Y decomposition).[15] Instead of using systematic CNOT gates,[17] non-parameterized random qubit interactions, known as the Transverse-field Ising model, could be employed.[15] However, the regression was unstable and sometimes failed, depending on the randomness of the qubit interactions with small circuit depth $m \leq 3$, which motivated us to use repeated CNOT gates (Figure S 1).

A QCL circuit with the qubit number of $n = 2$ and the depth of $m = 3$ was selected to learn the one-dimensional functions. Here, practically one-dimensional vector $\boldsymbol{x} = (x_1, x_1)$ was encoded in the two-qubit circuit, enabling the higher expressiveness of $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ (Figure S 1). The final output was scaled by a constant factor of two as a hyperparameter ($\hat{y} \mapsto 2\hat{y}$).[15, 17]

QCL circuit was able to learn versatile functions of $y = x$, $\sin(x/2)$, $\sin(x)$, and e^{x-1} (Figure 4a and Figure S 2). The machine learning model was trained with about six training datasets randomly sampled, and they predicted about ten random testing data. No significant prediction errors were detected (Figure 4a and Figure S 2).

The unique advantage of the QCL model is highlighted by comparing it with conventional regression models.[18] We examined standard algorithms of support vector machine for regression (SVR), random forest regression (RFR), Bayesian ridge regression (BYR), and Gaussian process regressor (GPR) (Figure 4a, Figure S 2, Figure S 3, Table S2).[5] SVR works on kernel trick,

enabling the learning of even non-linear functions. RFR is a standard ensemble model of decision trees. Its reliable prediction is frequently employed with experimental material databases.[5] BYR is a robust model for probabilistic linear regression, potentially strong against small datasets. GPR is similar to SVR, but its stochastic process can offer more flexible fitting with smaller datasets.

SVR, RFR, and BYR could not predict either $y = x$, $\sin(x)$, or e^{x-1}. Prediction in extrapolating regions, where x_i and $\hat{y}$ range out of the training datasets, was unsuccessful (Figure 4a and Figure S 2). The SVR model assuming a non-linear Gaussian kernel mimicked $\sin(x)$, but gave a bent curve in the untrained regions of $y = x$. RFR displayed similar responses to SVR, with the unfavorable step-wise prediction by the noncontinuous decision-tree algorithm. The linear BYR model was predictable of $y = x$, but never of $\sin(x)$. Even though their hyperparameters were changed, the three models could not predict the functions (Figure S 3). Due to the algorithm biases, many conventional supervised models could not switch linear and non-linear predictions.

We also examined more complex machine learning models, GPR and multilayer perceptron (MLP). GPR with radial basis function (RBF)-type kernels offered promising predictions due to their non-linear and stochastic algorithms (Figure 4a, Figure S 3).[18] MLPs with different activation functions (ReLu, tanh, and sigmoid) and hidden layer numbers (1 to 4) did not afford sufficient performances. The models could switch linear and non-linear functions, but larger errors were obtained because of too many trainable parameters against the inputted data

The performances of the regression models were validated by repeating the random data preparation and learning processes 30 times (Figure S 4, Table S3). On average, two GPR models with RBF or RBF + Dot-product kernels exhibited the smallest error. The following best was the

QCL regression: the model, capable of handling linear and non-linear interactions from small databases, offers a new option for solving regression tasks of various datasets.

4. Analyzing regression steps of QCL

Although QCL models work on complex prediction algorithms (e.g., Table S1), their estimation steps can be visualized (Figure S 5). As defined in **Eq 1**, a quantum state is expressed by a complex vector of $\boldsymbol{w} = (w_1, \dots, w_{2^n})^T$ $(w_i \in \mathbb{C})$. We showed the change of coordinates w_i in a simple example circuit of $U(\theta)V(x) = R_{1,y}(\theta) \cdot \mathrm{CNOT}_{1,2} \cdot R_{2,x}(x) \cdot R_{1,x}(x)$ $(\theta = 1.0, x = 0.6)$ for a two-qubit system (Figure S 5a). The applications of $R_{1,x}$ and $R_{2,x}$ gates changed w_1 to w_4 by shifting the complex coordinates of w_2, w_3, and w_4. Then, w_2 and w_4 were swapped by the $\mathrm{CNOT}_{2,1}$ gate. The final output of $\hat{y} = |w_1|^2 + |w_2|^2 - |w_3|^2 - |w_4|^2$ (**Eq 2**) was given followed by the rotation with $R_{1,y}$. Although state vectors are not observable in real world, state-vector simulation reveals that such rotation and swapping effects by quantum gates. Developing more sophisticated visualization methods (e.g., heatmap for neural networks)[11] is needed to provide deeper insights into more explainable QCL.

We calculated $|w_i|^2$ for the trained QCL models of $y = \sin(x)$ and x (Figure S 5, Figure 5b,c,d). Bending curves were observed for each term against x even with the linear function of $y = x$. This means that the QCL model worked through non-linear processes even with linear systems. The final output was made from the slight difference of $|w_1|^2 + |w_2|^2$ and $|w_3|^2 + |w_4|^2$.

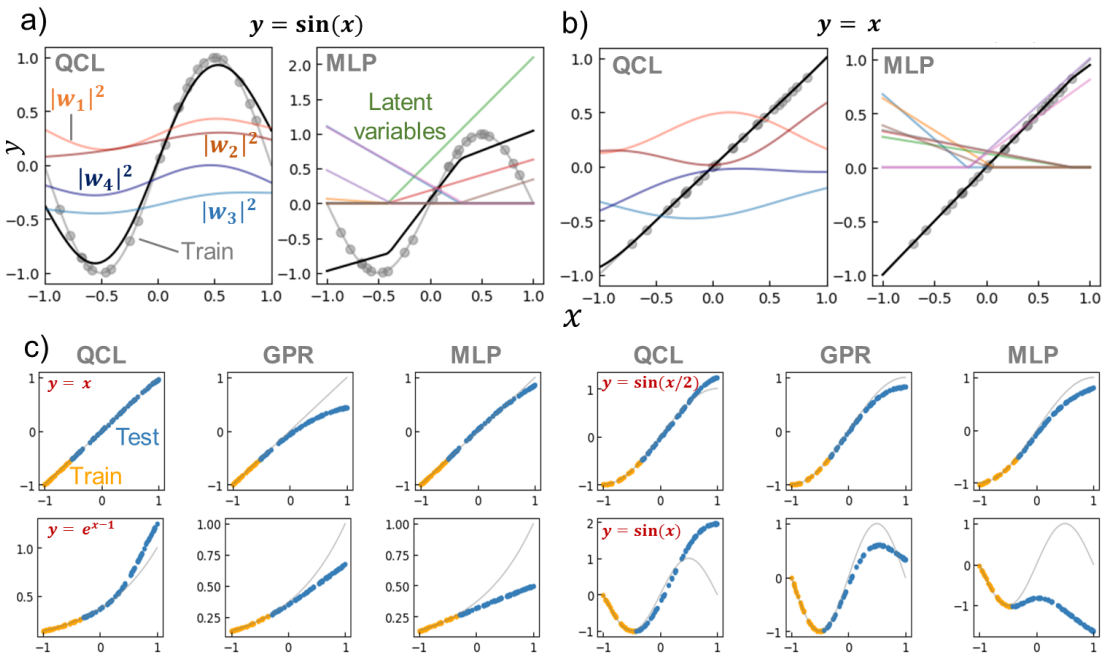


Figure 5 a) Prediction process of $y = \sin(x)$ by a trained QCL model ($m = 3, n = 2$) or multilayer perceptron (MLP, 8-dimensional hidden layer and activation function of ReLu). The model was trained with 24 random records (gray plots). Black lines show predictions, and colored represent latent variables. b) Prediction process for $y = x$. c) Extrapolating predictions by QCL, GPR (RBF), and MLP models. After randomly generating 100 points, 70% of the data with high y were selected as testing (extrapolating) data. Full results are shown in Figure S 7 and Figure S 8.

As a control for QCL, MLP regressors were examined for prediction. The MLP model contained one 8- (or 16-) dimensional hidden layer for prediction with the standard non-linear activation function of ReLu or tanh.[29] The overall design of QCL and MLP is slightly similar (Figure S 5b). Both models encode $\boldsymbol{x}$ to the first latent vector $\boldsymbol{w}'$, convert into another state of $\boldsymbol{w}$ by $\boldsymbol{\theta}$–dependent transformation, and finally calculate $\hat{y}$ from $\boldsymbol{w}$. The main differences are a) QCL maintains complex latent variables, whereas MLP usually has real numbers, and b) only linear (or more precisely, unitary) transformation is available by QCL during the conversion of $\boldsymbol{w}'$ to $\boldsymbol{w}$.

MLP models were not predictive of the one-dimensional functions with small training data (around 20 records, Figure 5a, Figure S 6). A simple formulal of $y = \sin(x)$ could not be fitted by MLP, even through different hyperparameters were employed (hidden layer sizes of 8 or 16 and activation functions of ReLu or tanh, Figure S 6). The simplest $y = x$ was successful, yet $y = \sin(x/2)$ and e^{x-1} were partially failing. Although complexing the circuit design, deep learning, will enhance fitting results, it also induces overfitting problems and requires larger datasets.

5. Keys to extrapolation by QCL

QCL affords promising prediction performances in extrapolating regions. We try to shed light on the reason by clarifying the requirements for extrapolation. First, the regression algorithms must provide $\hat{y}$ outside the scope of trained y. Several models, such as decision trees and SVRs with RBF kernels, would not meet the criterion (Figure 4). Second, the algorithms have to mimic the original data trend. For a simple example case, a quadratic function can perfectly predict the responses of ideal free falling. On the other hand, nobody knows most systems' exact functions and explanatory parameters. Instead, researchers should provide appropriate algorithms and parameters from their domain knowledge to come close to the ground truth.

The QCL model is specialized in mimicking the gently-sloping, non-linear functions. The model gave better performances in predicting linear, e^{x-1}, and $\sin(x/2)$ functions compared to GPR and MLP (Figure 5, Figure S 7, Figure S 8). The prediction error did not change drastically even though the extrapolation ratio in the dataset was changed from 10% to 90%, whereas others did (Figure S 8). On the other hand, poorer results were obtained with steeper functions, $\sin(x)$ and $\sin(2x)$. The QCL model with the current parameter was unable to fit $\sin(2x)$ ($n = 2, m =$

3, Figure S 2). The results indicated that the current QCL model was specialized in predicting gently-sloping, non-linear functions.

The smooth characteristics of the QCL model originated from the small trainable parameters and the regularization effect of quantum gates. The model had much smaller trainable parameters than MLP. The dimension of $\boldsymbol{\theta}$ for QCL ($n = 2, m = 3$) was only 15, whereas 27 and 51 parameters were needed even for the unsuccessful MLP models (with hidden layer dimensions of 8 and 16, respectively). The smaller trainable parameters and continuous sinusoidal basis resulted in smooth curves. Further, the unitary restriction of $|w_1|^2 + |w_2|^2 + |w_3|^2 + |w_4|^2 = 1$ should also have suppressed the outlier prediction as the regularization.

In summary, the current QCL model has a chance to outperform conventional linear and non-linear regression algorithms when smooth curves are supposed with the original datasets. Although the actual chemical and material systems do not always meet the requirement, the success in sparse (linear) modeling[5, 11] encourages researchers to expand the idea to smooth non-linear functions by QCL or other algorithms. Further tuning of QCL models will also offer capabilities of more complex functions, which should be examined in future research.

6. Prediction by an actual quantum computer

The biggest problem of QCL is the long calculation time with large qubit systems. The training time increased exponentially against the number of qubits n, and became significant (> 10^2 seconds) around $n = 8$ even with one-dimensional datasets (Figure S 9).

Instead of calculating state vectors by **Eq 1**, prediction can also be made by observing the actual quantum system's eigenvalues: this is the real QCL (**Eq 7**). Calculation cost will not increase exponentially because nature automatically does the calculation according to quantum mechanics.

The probabilistic sampling was examined with an IBM quantum computing machine (Figure S 10). The model was trained via the state-vector method. Then, we calculated statistical probabilities of upward (↑) or downward (↓) eigenvalues ($p_\uparrow$ or $p_\downarrow$) from the quantum system to predict $\hat{y} = p_\uparrow - p_\downarrow$ (**Eq 7**).

Quantum sampling suffered from more significant prediction errors than the classical state-vector calculation. Mean squared error (MSE) for the training dataset of $y = \sin(x)$, with a circuit of qubit number $n = 2$ and depth $m = 2$, was 0.0007 and 0.15 for state-vector and quantum sampling methods, respectively. When the circuit depth was increased to 3 or 4, the predicted values did not look like the original trigonometric curves. The errors were mainly caused by the computational noise of the quantum system (Figure S 10).[19] For practical usage, the number of quantum gates in the circuit must be reduced to suppress the effects of noise. More facile access to quantum machines is also essential because calculation takes about $10^1 - 10^3$ seconds to predict just one record by the heavily crowded cloud system. The superparallel advantage of quantum machines for QCL will be achieved when the computers can handle large qubit numbers n (>> 10) with negligible noise and prompt server responses.

Apart from hardware, the development of theoretical approaches is also essential. For instance, QCL accepts the limited domain of $\hat{y}$ and x_i. The unitarity of operational matrices restricts the predicted value of $-1 \leq \hat{y} \leq 1$. Although not mandatory, the explanatory variable x_i should range in $-\pi \leq x_i \leq \pi$ owing to the periodicity of trigonometric functions in rotational gates (**Eq 4**, **Eq**

5). For practical regression tasks, linear or non-linear conversion may be needed, whereas x_i and y were set in $[-1, +1]$ in this theoretical study (e.g., use of sigmoid: $1/(1+e^{-x})$ and logit: $\log(\hat{y}/(1-\hat{y}))$).

7. Predicting molecular properties

We examined the QCL models to predict experimental molecular properties from their structures. Four standard experimental databases of a) log solubility in water (estimating the aqueous solubility: ESOL),[9] b) melting point (Jean-Claude Bradley open melting point dataset),[30] c) octanol/water distribution coefficient (lipophilicity: Lipo),[9] and d) hydration free energy of small molecules in water (Solv)[9] were selected as the benchmark. Regardless of the qubit limitations, the benefit of QCL was observed with actual materials- or chemo-informatics tasks.

As explanatory variables, molecular features in the databases were calculated by a conventional ca. 200-dimensional descriptor algorithm of RDKit.[31] The method can facilely quantify molecular characteristics by various indicators, such as molecular weight and the number of specific atoms in a molecule. Due to the high calculation cost of QCL, the descriptors were compressed to an 8-dimensional vector by principal component analysis.[32] All explanatory and target variables were normalized in $[-1, +1]$.

Small datasets were prepared artificially, assuming the actual materials informatics projects. From the master databases, 8, 16, 32, 64, 128, 256, or 512 records were sampled randomly. Then, the top 20% records of y were extracted as the testing data: these were model tasks for extrapolating regression. The random selection and regression tasks were repeated 2000/(dataset size) times for statistical verification (Figure S 11).

QCL improved prediction performance more than conventional models with several conditions. For instance, QCL exhibited the smallest MSE of 0.25 for the testing data, with the melting point database of random 64 records (Figure 6a). Larger errors were observed with other models (RFR: 0.35, SVR: 0.30, BYR: 0.57, GPR: 0.61). Most of $\hat{y}$ by RFR and SVR ranged in the region of only trained y, meaning that extrapolation was unsuccessful, due to their decision-tree and radical basis kernel-based algorithms.[2]

Linear-compatible models of BYR and GPR made some extrapolating predictions, exceeding the maximum y of training records (Figure 6a). However, the model underestimated several test cases, giving large MSEs of 0.57 and 0.61, respectively. Another linear regression algorithm, partial least squares regression (PLS), was also examined as a regular model for materials informatics.[33] Nevertheless, the model suffered from the largest MSE of 0.94. We doubt that the linear models could not faithfully catch up with the nonlinearity of the current experimental systems.

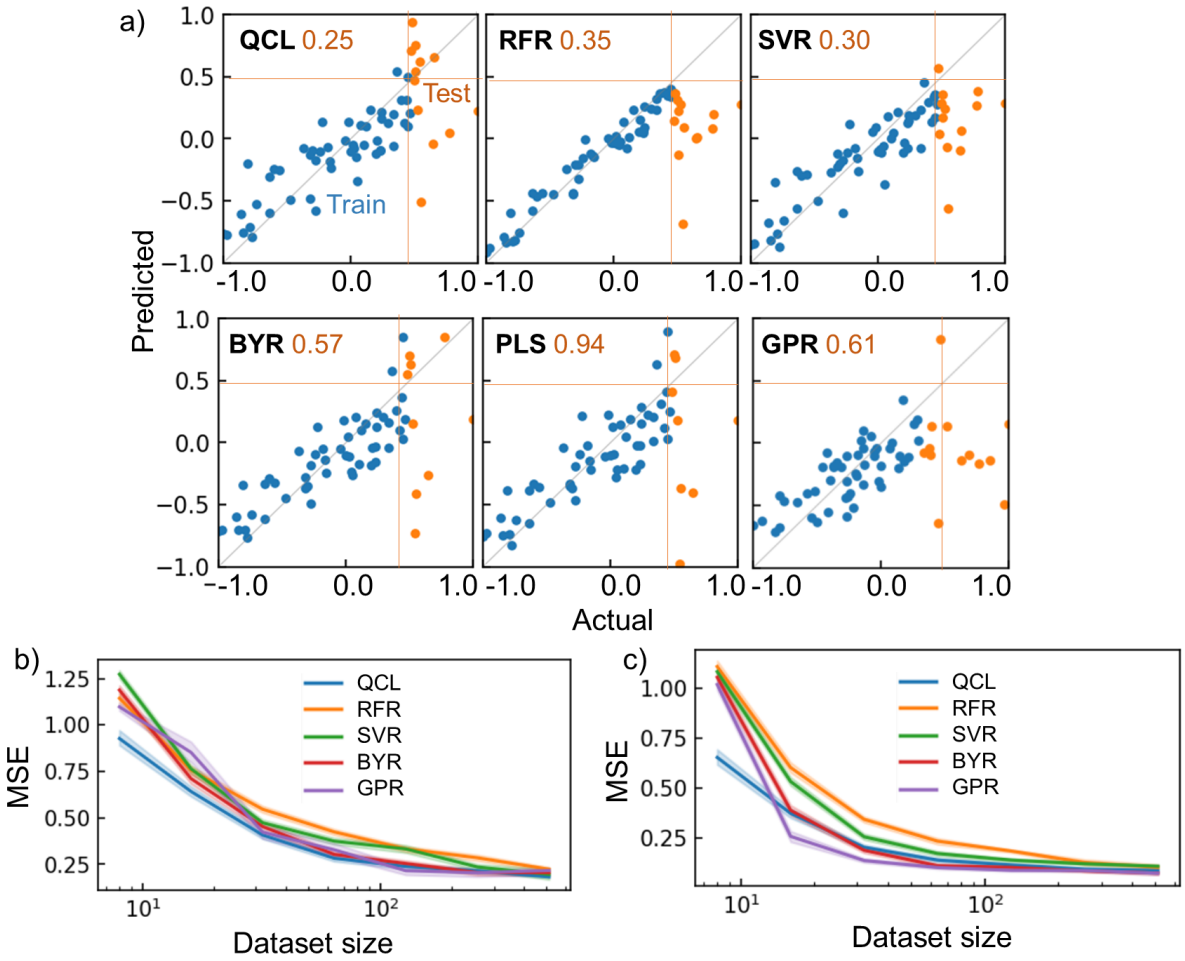


Figure 6 Regression results for chemical properties. a) Actual and predicted parameters for the melting point dataset, using QCL ($n = 8, m = 3$) and other regressors. Dataset size was 64. The top 20% of y records were extracted as testing data, whose MSE is shown as orange numbers. b) MSE for the regression tasks of melting point as a function of dataset size. Datasets were generated randomly and repeatedly. Transparent regions show standard errors with 68% confidence intervals. Results for PLS are not shown because the average MSE was too large. c) Results for ESOL. The results with other databases (Figure S 12) and results for interpolating tasks are shown in Figure S 13 and Figure S 14. RBF + dot-product was used for GPR.

The models' performances were examined by repeating the random shuffling and regressions (Figure 6b,c, Figure S 12). Up to the dataset size of 100, QCL almost displayed the smallest error of the models (Figure 6b). The quantum model was also robust against tiny datasets of ESOL and Solv (Figure 6c and Figure S 12). The QCL model also benefited from regular interpolating regression tasks, where 20% of testing data were sampled randomly (Figure S 13). The model

exhibited the best performance with the ESOL datasets, up to 32 records. Naturally, other models sometimes outperformed QCL under different conditions. There is no omnipotent algorithm applicable to any problem (no-free-lunch theorem).[34] More careful analysis of predicting processes for each case is needed to pursue better performances in future research.

Although we currently have no clear clue about the remarkable performances of QCL, the gently-sloping assumption of datasets might be a key to prediction. As demonstrated with the one-dimensional functions, the QCL model could fit linear and smooth curves (Figure 4). If the experimental molecular structure ($\boldsymbol{x}$)–property (y) relationships were not so fluctuated, their data trend could be mimicked by QCL. We are examining the data trends more carefully by considering the multivariable factors and distinguishing which functions are suitable for QCL.

A drawback of QCL for material exploration is the limited dimension of explanatory parameters. If conventional models conducted regressions without dimension reduction, they offered better performances than QCL (Figure S 12 and Figure S 13). From another perspective, however, we can understand that the still large prediction errors by QCL were soluble by expanding the dimension. Preliminarily selecting essential parameters by other methods, such as sparse modeling,[12] will also be critical to utilize QCL.

The encoding method of $\boldsymbol{x}$ to quantum circuits is also a challenge of QCL.[15-17, 26] The current model did not require two qubits for one variable of x_i, in contrast to one-dimensional regressions (Figure S 1). No significant improvement in prediction was detected even though the descriptors were compressed to 4-dimensional vectors and inputted to the 8-qubit model (i.e., $\boldsymbol{x} = (x_1, x_1, x_2, x_2, \dots, x_4, x_4)$, Figure S 14). The success may be explained by exponential natures of the state vectors (i.e., 2^n-dimensional vectors and fully connected interactions). Encoding multiple

values to one qubit is gradually becoming possible,[26, 28] whose circuit optimization will also increase the dimensions of explanatory parameters.

Conclusions

We examined the fundamental steps of quantum circuit learning (QCL) for regression and prediction performances of experimental molecular properties. The superparallel and entanglement natures of the quantum systems led to the exponential increase of model complexity along with qubit numbers. Our study showed that the unitarity quantum operations contributed to the flexible fitting of linear and smooth non-linear functions, even from small datasets and extrapolating regions. QCL models had a chance to outperform conventional models with several experimental molecule databases. The long simulation time of QCL by classical computers is a challenge for practical applications. However, it is intrinsically soluble by developing algorithms and hardware. Supercomputers can now handle over 30 qubits,[35, 36] whose simulations will hint us at a more efficient way of calculations. We are also continuing to examine the potential of QCL with various material and chemical prediction tasks.

Experimental section

Data and Software Availability

Programs and databases used in this study are available as open access via GitHub (https://github.com/KanHatakeyama/qcl)

Deriving equations for a regression model of QCL

A regression model **Eq 2** can be derived according to quantum mechanics and computing theory.[28] A state of one qubit is typically expressed with a linear combination of two basis vectors, $|0\rangle$ and $|1\rangle$ (**Eq 8**).

	$$\|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$$ $$\|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$$	**Eq 8**

For n-qubit system, 2^n-dimensional bases are needed to describe an arbitrary quantum state because of quantum entanglement.[28] Their bases can be made by the tensor products of $|0\rangle$ and $|1\rangle$ (**Eq 9**).

	$$\|\psi\rangle = \boldsymbol{w} = w_1\|00\rangle + w_2\|01\rangle + w_3\|10\rangle + w_4\|11\rangle$$ $$\|00\rangle = \|0\rangle\otimes\|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\times 1\\1\times 0\\0\times 1\\0\times 0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}$$ $$\|01\rangle = \|0\rangle\otimes\|1\rangle = (0 \quad 1 \quad 0 \quad 0)^T$$ $$\|10\rangle = \|1\rangle\otimes\|0\rangle = (0 \quad 0 \quad 1 \quad 0)^T$$ $$\|11\rangle = \|1\rangle\otimes\|1\rangle = (0 \quad 0 \quad 0 \quad 1)^T$$ (these are examples for $n = 2$)	**Eq 9**

By quantum computing, the initial state of $|0 \ldots 0\rangle = |0\rangle\otimes|0\rangle\otimes \ldots \otimes|0\rangle = (1 \quad 0 \quad \ldots \quad 0)^T$ is transformed into another form of $|\psi\rangle$ by repeated application of quantum gates (e.g., R_x, R_y, and CNOT), as unitary matrices (**Eq 1**, **Eq 3**). In actual quantum systems, the state vector $|\psi\rangle$ itself cannot be observed, but expected values of some Hermitian operators are observable. Although there are many observation ways, the most straightforward and popular operation is to detect

upward (↑) or downward (↓) eigenvectors for one qubit against the z-axis (**Eq 7**).[15, 28] Its mathematical expression can be given by applying a Pauli's Z operator to the first qubit in a circuit (**Eq 10**).

	$\hat{y} = \langle Z_1 \rangle = \langle \psi \vert Z_1 \vert \psi \rangle = (w_1^* \quad \dots \quad w_{2^n}^*) Z_1 \begin{pmatrix} w_1 \\ \vdots \\ w_{2^n} \end{pmatrix} =$ **Eq 2** $Z_1 = Z \otimes I_2 \otimes I_2 \otimes \dots \otimes I_2$ (repeat 2^n-1 times) $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	**Eq 10**

Calculation environment

Calculations were conducted with a Python 3 environment. Quantum computing libraries of Qulacs and Qiskit were introduced for simulation and actual quantum computing. Conventional regression models were made with scikit-learn.[32] Analytical solutions were calculated with Sympy.[37] Computations were done with a standard workstation without using graphics processing units (Intel Core i9-9900K CPU @ 3.60 GHz, 32 GB memory, and Ubuntu 16.04 operating system).

Preparation of regression models

Unless noted otherwise, prediction models were generated with the configuration of Figure 3b and state vector calculation using classical computers. The number of trainable parameters (or dimension of $\boldsymbol{\theta}$) were $3mn - 3m(n-1)$. The latter term comes from the fact that rotation of i-th ($i > 1$) qubit in the final depth layer does not affect the state of the 1st qubit for observation. Calculations were done using a Qulacs library.[38] Final prediction was made using a constant factor of two as a hyperparameter ($\hat{y} \mapsto 2\hat{y}$). Models were trained so that the summation of squared

prediction errors $(y - \hat{y})^2$ for training data became smaller. Basin-hopping solver implemented in Scipy library[39] was employed to optimize $\boldsymbol{\theta}$. Ising model Hamiltonian was also examined during circuit optimization instead of CNOT gates (Figure S 1). According to the method of Mitarai et al., Hamiltonian was made randomly, and its time evolution operator was generated via Suzuki-Trotter transformation (time step of 0.1).[15]

For actual quantum computation and its simulation, predictions were conducted using a Qiskit library.[40] For higher accuracy, the training parameter $\boldsymbol{\theta}$ was preliminarily set by the state vector calculation by Qulacs. During prediction, sampling was repeated 1000 times for one record to obtain $p_\uparrow$ and $p_\downarrow$. Cloud service of IBM Quantum systems was used for quantum computing. Five-qubit systems were employed for sampling (mainly using a machine named ibmq_quito, having a quantum volume of 16 and clops of 2.5K).

The following conventional models were introduced using the scikit-learn module: support vector machine for regression (SVR, radial basis function kernel), random forest regression (RFR, 100 decision trees), Bayesian ridge regression (BYR), Gaussian process regressor (GPR), partial least squares (PLS) regression (default dimension of 8). GPR models were constructed using some selected kernel plus a white kernel. Unless noted otherwise, default hyperparameters were used. MLP models were prepared using a Keras library.[41] The model had a one-dimensional inputting layer, one (or multiple) 8- or 16-dimensional hidden layer(s), and a one-dimensional output layer (multiple hidden layers were examined in Figure S 3 and Figure S 4). Relu, sigmoid, or tanh activation functions were introduced in the model. All training data (24 records) were simultaneously inputted into the model, using MSE loss and Adam optimizer. Due to the limited records, training was systematically repeated for 1000 epochs without making validation datasets.

Regression of one-dimensional functions

In Figure 4 and related figures, regression models were examined with one-dimensional functions of $y = x$, $\sin(ax)$, e^{x-1} (a: $const.$) with $-1 \leq x, y \leq +1$. Random $N = 20$ records were sampled to prepare a master dataset. For simplicity, no noise was added to the dataset. Then, the top 10%, lowest 10%, and other random 20% of the records were extracted as the testing data. The rest 60% was used for training. In Figure S 9, N was set to be 50 for more accurate calculation time estimation. In Figure S 10, $N = 10$ was used because of a long access time to connect the IBM cloud server. For extrapolation tasks, random $N = 100$ records were generated and 10, 30, 50, 70, or 90% of the data were used for testing (Figure S 12).

Chemical property regression

Four types of standard experimental molecular databases were introduced: a) log solubility in water (estimating the aqueous solubility: ESOL), b) melting point (Jean-Claude Bradley open melting point dataset), c) octanol/water distribution coefficient (lipophilicity: Lipo), and d) hydration free energy of small molecules in water (Solv).[9] Features of molecules were quantified as about 200-dimensional molecular descriptors by an RDKit library (Table S4). Then, the descriptors were compressed to 8-dimensional vectors by principal component analysis by a scikit-learn module.[32] In Figure S 14, descriptors were compressed to 4-dimensional vectors instead.

For regression, quantum circuit models with $n = 8$ and $m = 3$ were employed. Datasets were prepared by randomly sampling 8, 16, 32, 64, 128, 256, or 512 records from the master databases (Figure S 11). All variables (y, x_i) in each dataset were normalized in the range of $[-1, +1]$. As testing data, 20% of the top y records were extracted in Figure 6 and Figure S 12. Random 20% data were extracted for testing in Figure S 13 and Figure S 14. The random dataset generation and

prediction processes were repeated 2000/(dataset size) times for statistical verification. The figures display the test data's mean squared error (MSE) as box plots. The maximum *y*-axis was set to be 4 for easier understanding (excessive outliers are not shown in the graphs). Unless noted otherwise, default hyperparameters of the scikit-learn library were used for the conventional models.

AUTHOR INFORMATION

Corresponding Author

*Kan Hatakeyama-Sato (satokan@toki.waseda.jp)

*Kenichi Oyaizu (oyaizu@waseda,jp)

Author Contributions

All authors have given approval to the final version of the manuscript.

Funding Sources

This work was partially supported by Grants-in-Aid for Scientific Research (Nos. 21H04695, 18H05515, 20H05298, 22H04623, and 21H02017) from MEXT, Japan. The work was partially supported by JST FOREST Program (Grant Number JPMJFR213V, Japan) and the Research Institute for Science and Engineering, Waseda University

REFERENCES

1. S. Hong, C. H. Liow, J. M. Yuk, H. R. Byon, Y. Yang, E. Cho, J. Yeom, G. Park, H. Kang, S. Kim, Y. Shim, M. Na, C. Jeong, G. Hwang, H. Kim, H. Kim, S. Eom, S. Cho, H. Jun, Y. Lee, A. Baucour, K. Bang, M. Kim, S. Yun, J. Ryu, Y. Han, A. Jetybayeva, P.

P. Choi, J. C. Agar, S. V. Kalinin, P. W. Voorhees, P. Littlewood and H. M. Lee, *ACS Nano*, 2021, **15**, 3971-3995.
2. J. Schmidt, M. R. G. Marques, S. Botti and M. A. L. Marques, *Npj Comput. Mater.*, 2019, **5**, 83.
3. M. Scheffler, M. Aeschlimann, M. Albrecht, T. Bereau, H. J. Bungartz, C. Felser, M. Greiner, A. Gross, C. T. Koch, K. Kremer, W. E. Nagel, M. Scheidgen, C. Woll and C. Draxl, *Nature*, 2022, **604**, 635-642.
4. Y. C. Lo, S. E. Rensi, W. Torng and R. B. Altman, *Drug Discov Today*, 2018, **23**, 1538-1546.
5. R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi and C. Kim, *Npj Comput. Mater.*, 2017, **3**, 54.
6. Y. LeCun, Y. Bengio and G. Hinton, *Nature*, 2015, **521**, 436-444.
7. K. Hatakeyama-Sato and K. Oyaizu, *Commun. Mater.*, 2020, **1**, article number: 49.
8. K. Hatakeyama-Sato, T. Tezuka, M. Umeki and K. Oyaizu, *J. Am. Chem. Soc.*, 2020, **142**, 3301-3305.
9. Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing and V. Pande, *Chem. Sci.*, 2018, **9**, 513-530.
10. S. Nagasawa, E. Al-Naamani and A. Saeki, *J. Phys. Chem. Lett.*, 2018, **9**, 2639-2646.
11. H. Yamada, C. Liu, S. Wu, Y. Koyama, S. Ju, J. Shiomi, J. Morikawa and R. Yoshida, *ACS Cent. Sci.*, 2019, **5**, 1717-1730.
12. R. Mizuguchi, Y. Igarashi, H. Imai and Y. Oaki, *Nanoscale*, 2021, **13**, 3853-3859.
13. J. Mairal, F. Bach and J. Ponce, 2014, **arXiv:1411.3230**.
14. H. Numazawa, Y. Igarashi, K. Sato, H. Imai and Y. Oaki, *Adv. Theory Simul.*, 2019, **2**, 1900130.
15. K. Mitarai, M. Negoro, M. Kitagawa and K. Fujii, *Physical Review A*, 2018, **98**, 032309.
16. J. G. Liu and L. Wang, *Physical Review A*, 2018, **98**, 062324.
17. T. Suzuki and M. Katouda, *J. Phys. Commun.*, 2020, **4**, 125012.
18. T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning*, Springer New York, NY, 2009.
19. K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek and A. Aspuru-Guzik, *Rev. Mod. Phys.*, 2022, **94**, 015004.
20. J. Preskill, *Quantum*, 2018, **2**, 79.
21. Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferova, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. D. Sawaya, S. Sim, L. Veis and A. Aspuru-Guzik, *Chem. Rev.*, 2019, **119**, 10856-10915.
22. I. G. Ryabinkin, T. C. Yen, S. N. Genin and A. F. Izmaylov, *J. Chem. Theory Comput.*, 2018, **14**, 6317-6326.
23. R. Xia and S. Kais, *Nat Commun*, 2018, **9**, 4195.
24. J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe and S. Lloyd, *Nature*, 2017, **549**, 195-202.
25. S. A. Stein, R. L'Abbate, W. Mu, Y. Liu, B. Baheri, Y. Mao, Q. Guan, A. Li and B. Fang, 2021, **arXiv:2012.00256**.
26. M. Schuld, A. Bocharov, K. M. Svore and N. Wiebe, *Physical Review A*, 2020, **101**, 032308.

27. Y. Takaki, K. Mitarai, M. Negoro, K. Fujii and M. Kitagawa, *Physical Review A*, 2021, **103**.
28. M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge: Cambridge University Press, 2010.
29. L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaria, M. A. Fadhel, M. Al-Amidie and L. Farhan, *J Big Data*, 2021, **8**, 53.
30. Jean-Claude Bradley Open Melting Point Dataset (2014). DOI: 10.6084/m9.figshare.1031637
31. RDKit: Open-source cheminformatics; , http://www.rdkit.org).
32. 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 E. Duchesnay, *J. Mach. Learn. Res.*, 2011, **12**, 2825-2830.
33. K. Rajan, *Mater. Today*, 2005, **8**, 38-45.
34. D. H. Wolpert and W. G. Macready, *IEEE T. Evolut. Compt.*, 1997, **1**, 67-82.
35. S. Imamura, M. Yamazaki, T. Honda, A. Kasagi, A. Tabuchi, H. Nakao, N. Fukumoto and K. Nakashima, *arXiv:2203.16044*, 2022.
36. Z. Wang, Z. Chen, S. Wang, W. Li, Y. Gu, G. Guo and Z. Wei, *Sci Rep*, 2021, **11**, 355.
37. A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, Š. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman and A. Scopatz, *PeerJ Comput. Sci.*, 2017, **3**.
38. Y. Suzuki, Y. Kawase, Y. Masumura, Y. Hiraga, M. Nakadai, J. Chen, K. M. Nakanishi, K. Mitarai, R. Imai, S. Tamiya, T. Yamamoto, T. Yan, T. Kawakubo, Y. O. Nakagawa, Y. Ibe, Y. Zhang, H. Yamashita, H. Yoshimura, A. Hayashi and K. Fujii, *Quantum*, 2021, **5**, 559.
39. P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, I. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt and C. SciPy, *Nat Methods*, 2020, **17**, 261-272.
40. Qiskit: An Open-source Framework for Quantum Computing, https://zenodo.org/record/6476573, DOI: 10.5281/zenodo.2573505).
41. F. Chollet, Keras, https://github.com/fchollet/keras).

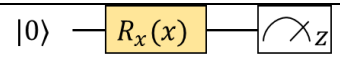
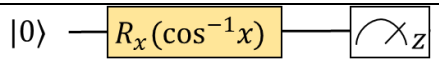
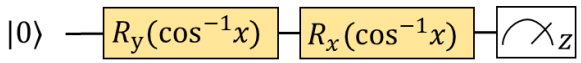
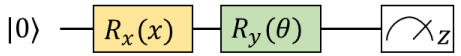
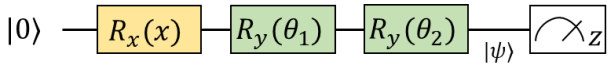
Supplementary Information

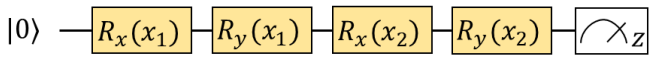
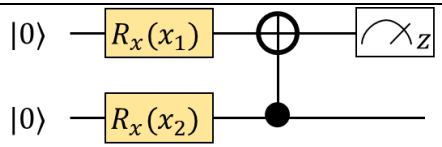
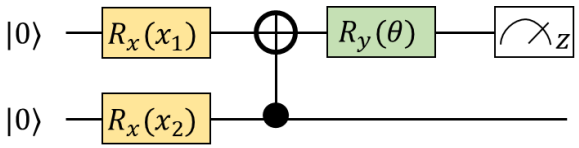
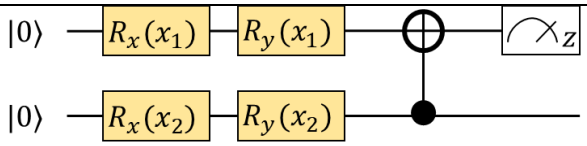
Quantum circuit learning to predict experimental chemical properties

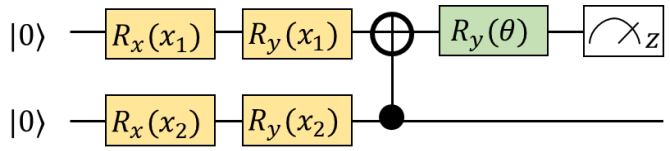
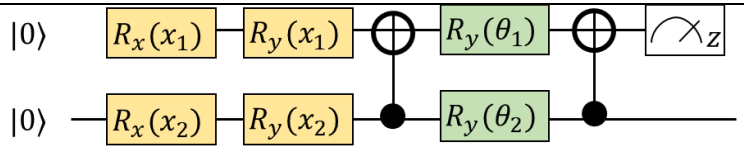
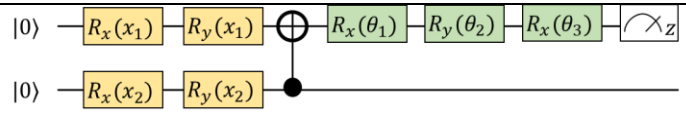
Kan Hatakeyama-Sato,* Yasuhiko Igarashi,
Takahiro Kashikawa, Koichi Kimura, Kenichi Oyaizu*

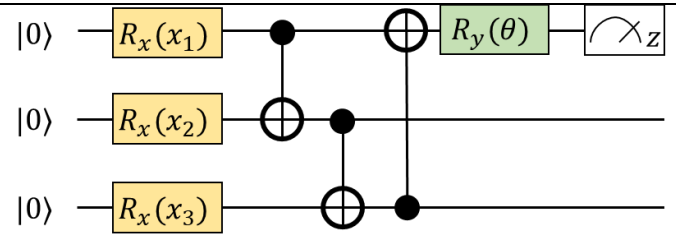
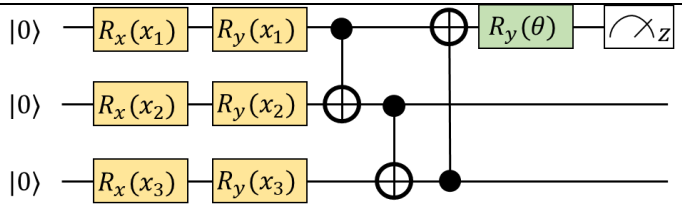
Table S1 Examples of simple quantum circuits and mathematical expressions.

NOTE: for simple expression, the results for $R_x(2t)$ and $R_y(2t)$ are used for $\hat{y}$ ($t \mapsto 2t$ in **Eq 4** and **Eq 5**).

Circuit	V	U	$\hat{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x})$
	$R_x(x)$	-	$\cos(2x)$
	$R_x(\cos^{-1}x)$	-	$2x^2 - 1$
	$R_x(\cos^{-1}x)$ $R_y(\cos^{-1}x)$	-	$4x^4 - 4x^2 - 1$
	$R_x(x)$	$R_y(\theta)$	$\frac{\cos(2\theta - 2x)}{2} + \frac{\cos(2\theta + 2x)}{2}$
	$R_x(x)$	$R_y(\theta_2)R_y(\theta_1)$	$-\frac{\cos(2\theta_2 - 2x)}{2} + \frac{\cos(2\theta_2 + 2x)}{2} + \frac{\cos(-2\theta_1 + 2\theta_1 + 2x)}{4} + \frac{\cos(2\theta_1 - 2\theta_1 + 2x)}{4} + \frac{\cos(2\theta_1 + 2\theta_1 - 2x)}{4} + \frac{\cos(2\theta_1 + 2\theta_1 + 2x)}{4}$

	$R_y(x_2)R_x(x_2)$ $\cdot R_y(x_1)R_x(x_1)$	-	$\frac{\cos(4x_1)}{4}+\frac{\cos(4x_2)}{4}$ $-\frac{\cos(2x_1-4x_2)}{4}$ $+\frac{\cos(2x_1+4x_2)}{4}$ $+\frac{\cos(4x_1-4x_2)}{8}$ $-\frac{\cos(4x_1-2x_2)}{4}$ $+\frac{\cos(4x_1+2x_2)}{4}$ $+\frac{\cos(4x_1+4x_2)}{8}+\frac{1}{4}$
	$R_x(x_1)\otimes R_x(x_2)$	CNOT(2,1)	$\frac{\cos(2x_1-2x_2)}{2}$ $+\frac{\cos(2x_1+2x_2)}{2}$
	$R_x(x_1)\otimes R_x(x_2)$	$(R_y(\theta)\otimes I_2)$ $\cdot$ CNOT(2,1)	$\frac{\cos(-2\theta+2x_1+2x_2)}{4}$ $+\frac{\cos(2\theta-2x_1+2x_2)}{4}$ $+\frac{\cos(2\theta+2x_1-2x_2)}{4}$ $+\frac{\cos(2\theta+2x_1+2x_2)}{4}$
	$(R_y(x_1)R_x(x_1))$ $\otimes($ $R_y(x_2)R_x(x_2))$	CNOT(2,1)	$(2\sin^2(x_1)-1)^2\,(2\sin^2(x_2)$ $-1)^2$

	$(R_y(x_1)R_x(x_1))$ $\otimes($ $R_y(x_2)R_x(x_2))$	$(R_y(\theta)\otimes I_2)$ $\cdot$ CNOT(2,1)	$\frac{(1-\cos(2x_2))^2\cos(2\theta)}{2}$ $+\frac{(1-\cos(2x_2))^2\cos(2\theta-4x}{4}$ $+\frac{(1-\cos(2x_2))^2\cos(2\theta+4x}{4}$ $-\frac{\cos(2\theta)}{2}-\frac{\cos(2\theta-4x_1)}{2}$ $+\frac{\cos(2\theta-2x_2)}{2}$ $+\frac{\cos(2\theta+2x_2)}{2}$ $+\frac{\cos(-2\theta+4x_1+2x_2)}{4}$ $+\frac{\cos(2\theta-4x_1+2x_2)}{4}$ $+\frac{\cos(2\theta+4x_1-2x_2)}{4}$ $+\frac{\cos(2\theta+4x_1+2x_2)}{4}$
	$(R_y(x_1)R_x(x_1))$ $\otimes($ $R_y(x_2)R_x(x_2))$	CNOT(2,1) $\cdot (R_y(\theta_1)\otimes R_y(\theta_2))$ $\cdot$ CNOT(2,1)	Eq S1
	$(R_y(x_1)R_x(x_1))$ $\otimes($ $R_y(x_2)R_x(x_2))$	$\cdot (R_x(\theta_3)\otimes I_2)$ $\cdot (R_y(\theta_2)\otimes I_2)$ $\cdot (R_x(\theta_1)\otimes I_2)$ $\cdot$ CNOT(2,1)	Eq S2

	$R_x(x_1)$ $\otimes R_x(x_2) \otimes R_x(x_3)$	$(R_y(\theta) \otimes I_2 \otimes I_2)$ · CNOT(3,1) · CNOT(2,3) · CNOT(1,2)	Eq S3
	$(R_y(x_1)R_x(x_1))$ $\otimes($ $R_y(x_2)R_x(x_2))$ $\otimes($ $R_y(x_3)R_x(x_3))$	$(R_y(\theta) \otimes I_2 \otimes I_2)$ · CNOT(3,1) · CNOT(2,3) · CNOT(1,2)	Eq S4

Eq S1

$$
\begin{aligned}
&(1-\cos(2x_1))^2\cos(2\theta_2) - \frac{(1-\cos(2x_2))^2\cos(-2\theta_1+2\theta_2+4x_1)}{8} + \frac{(1-\cos(2x_2))^2\cos(2\theta_1-2\theta_2+4x_1)}{8} - \frac{(1-\cos(2x_2))^2\cos(2\theta_1+2\theta_2-4x_1)}{8} + \frac{(1-\cos(2x_2))^2\cos(2\theta_1+2\theta_2+4x_1)}{8} \\
&+ \frac{\sin(-2\theta_1+2\theta_2+4x_2)}{8} + \frac{\sin(2\theta_1-2\theta_2+4x_2)}{8} + \frac{\sin(2\theta_1+2\theta_2-4x_2)}{8} - \frac{\sin(2\theta_1+2\theta_2+4x_2)}{8} - \frac{\sin(-2\theta_1+2\theta_2+2x_1+2x_2)}{8} - \frac{\sin(2\theta_1-2\theta_2-2x_1+2x_2)}{8} - \frac{\sin(2\theta_1-2\theta_2+2x_1-2x_2)}{8} \\
&+ \frac{\sin(2\theta_1-2\theta_2+2x_1+2x_2)}{8} - \frac{\sin(2\theta_1+2\theta_2-2x_1-2x_2)}{8} + \frac{\sin(2\theta_1+2\theta_2-2x_1+2x_2)}{8} + \frac{\sin(2\theta_1+2\theta_2+2x_1-2x_2)}{8} - \frac{\sin(2\theta_1+2\theta_2+2x_1+2x_2)}{8} - \frac{3\cos(2\theta_2)}{2} + \frac{\cos(2\theta_1-2\theta_2)}{4} + \frac{\cos(2\theta_1+2\theta_2)}{4} \\
&- \frac{\cos(2\theta_2-4x_1)}{4} + \cos(2\theta_2-2x_1) + \cos(2\theta_2+2x_1) - \frac{\cos(2\theta_2+4x_1)}{4} + \frac{\cos(-2\theta_1+2\theta_2+4x_1)}{4} + \frac{\cos(2\theta_1+2\theta_2-4x_1)}{4} - \frac{\cos(-2\theta_1+2\theta_2+4x_1+2x_2)}{8} - \frac{\cos(2\theta_1-2\theta_2-4x_1+2x_2)}{8} \\
&+ \frac{\cos(2\theta_1-2\theta_2+4x_1-2x_2)}{8} + \frac{\cos(2\theta_1-2\theta_2+4x_1+2x_2)}{8} - \frac{\cos(2\theta_1+2\theta_2-4x_1-2x_2)}{8} - \frac{\cos(2\theta_1+2\theta_2-4x_1+2x_2)}{8} + \frac{\cos(2\theta_1+2\theta_2+4x_1-2x_2)}{8} + \frac{\cos(2\theta_1+2\theta_2+4x_1+2x_2)}{8}
\end{aligned}
$$

Eq S2

$$
\begin{aligned}
&- \frac{(1-\cos(-2\theta_1+2\theta_2+2\theta_3))(\cos(4x_2)+1)}{32} - \frac{(1-\cos(2\theta_1-2\theta_2+2\theta_3))(\cos(4x_2)+1)}{32} - \frac{(1-\cos(2\theta_1+2\theta_2-2\theta_3))(\cos(4x_2)+1)}{32} - \frac{(1-\cos(2\theta_1+2\theta_2+2\theta_3))(\cos(4x_2)+1)}{32} \\
&- \frac{(1-\cos(-2\theta_1+2\theta_2+2\theta_3+4x_1))(\cos(4x_2)+1)}{64} - \frac{(1-\cos(2\theta_1-2\theta_2-2\theta_3+4x_1))(\cos(4x_2)+1)}{64} - \frac{(1-\cos(2\theta_1-2\theta_2+2\theta_3-4x_1))(\cos(4x_2)+1)}{64} - \frac{(1-\cos(2\theta_1-2\theta_2+2\theta_3+4x_1))(\cos(4x_2)+1)}{64} \\
&- \frac{(1-\cos(2\theta_1+2\theta_2-2\theta_3-4x_1))(\cos(4x_2)+1)}{64} - \frac{(1-\cos(2\theta_1+2\theta_2-2\theta_3+4x_1))(\cos(4x_2)+1)}{64} - \frac{(1-\cos(2\theta_1+2\theta_2+2\theta_3-4x_1))(\cos(4x_2)+1)}{64} - \frac{(1-\cos(2\theta_1+2\theta_2+2\theta_3+4x_1))(\cos(4x_2)+1)}{64} \\
&+ \frac{\cos(4x_2)}{4} - \frac{\cos(2\theta_1-2\theta_3)}{8} + \frac{\cos(2\theta_1+2\theta_3)}{8} + \frac{\cos(-2\theta_1+2\theta_2+2\theta_3)}{32} + \frac{\cos(-2\theta_1+2\theta_3+2x_1)}{8} - \frac{\cos(-2\theta_1+2\theta_3+4x_1)}{16} - \frac{\cos(-2\theta_1+2\theta_3+4x_2)}{16} + \frac{\cos(2\theta_1-2\theta_2+2\theta_3)}{32} + \frac{\cos(2\theta_1+2\theta_2-2\theta_3)}{32} \\
&+ \frac{\cos(2\theta_1+2\theta_2+2\theta_3)}{32} - \frac{\cos(2\theta_1-2\theta_3+2x_1)}{8} - \frac{\cos(2\theta_1-2\theta_3+4x_1)}{16} - \frac{\cos(2\theta_1-2\theta_3+4x_2)}{16} + \frac{\cos(2\theta_1+2\theta_3-4x_1)}{16} - \frac{\cos(2\theta_1+2\theta_3-2x_1)}{8} + \frac{\cos(2\theta_1+2\theta_3+2x_1)}{8} + \frac{\cos(2\theta_1+2\theta_3+4x_1)}{16} \\
&+ \frac{\cos(2\theta_1+2\theta_3-4x_2)}{16} + \frac{\cos(2\theta_1+2\theta_3+4x_2)}{16} - \frac{\cos(-2\theta_2+2\theta_3+4x_1)}{8} + \frac{\cos(2\theta_2-2\theta_3+4x_1)}{8} - \frac{\cos(2\theta_2+2\theta_3-4x_1)}{8} + \frac{\cos(2\theta_2+2\theta_3+4x_1)}{8} - \frac{\cos(-2\theta_1+2\theta_2+2\theta_3+2x_1)}{16} \\
&+ \frac{\cos(-2\theta_1+2\theta_2+2\theta_3+4x_1)}{64} + \frac{\cos(-2\theta_1+2\theta_2+2\theta_3+4x_2)}{64} + \frac{\cos(-2\theta_1+2\theta_3+2x_1+4x_2)}{16} - \frac{\cos(-2\theta_1+2\theta_3+4x_1+4x_2)}{32} + \frac{\cos(2\theta_1-2\theta_2-2\theta_3+2x_1)}{16} + \frac{\cos(2\theta_1-2\theta_2-2\theta_3+4x_1)}{64} \\
&+ \frac{\cos(2\theta_1-2\theta_2-2\theta_3+4x_2)}{64} + \frac{\cos(2\theta_1-2\theta_2+2\theta_3-4x_1)}{64} - \frac{\cos(2\theta_1-2\theta_2+2\theta_3-2x_1)}{16} + \frac{\cos(2\theta_1-2\theta_2+2\theta_3+2x_1)}{16} + \frac{\cos(2\theta_1-2\theta_2+2\theta_3+4x_1)}{64} + \frac{\cos(2\theta_1-2\theta_2+2\theta_3-4x_2)}{64} \\
&+ \frac{\cos(2\theta_1-2\theta_2+2\theta_3+4x_2)}{64} + \frac{\cos(2\theta_1+2\theta_2-2\theta_3-4x_1)}{64} - \frac{\cos(2\theta_1+2\theta_2-2\theta_3-2x_1)}{16} + \frac{\cos(2\theta_1+2\theta_2-2\theta_3+2x_1)}{16} + \frac{\cos(2\theta_1+2\theta_2-2\theta_3+4x_1)}{64} + \frac{\cos(2\theta_1+2\theta_2-2\theta_3-4x_2)}{64} \\
&+ \frac{\cos(2\theta_1+2\theta_2-2\theta_3+4x_2)}{64} + \frac{\cos(2\theta_1+2\theta_2+2\theta_3-4x_1)}{64} - \frac{\cos(2\theta_1+2\theta_2+2\theta_3-2x_1)}{16} + \frac{\cos(2\theta_1+2\theta_2+2\theta_3+2x_1)}{16} + \frac{\cos(2\theta_1+2\theta_2+2\theta_3+4x_1)}{64} + \frac{\cos(2\theta_1+2\theta_2+2\theta_3-4x_2)}{64} \\
&+ \frac{\cos(2\theta_1+2\theta_2+2\theta_3+4x_2)}{64} - \frac{\cos(2\theta_1-2\theta_3-4x_1+4x_2)}{32} + \frac{\cos(2\theta_1-2\theta_3-2x_1+4x_2)}{16} - \frac{\cos(2\theta_1-2\theta_3+2x_1-4x_2)}{16} - \frac{\cos(2\theta_1-2\theta_3+2x_1+4x_2)}{16} - \frac{\cos(2\theta_1-2\theta_3+4x_1-4x_2)}{32} \\
&- \frac{\cos(2\theta_1-2\theta_3+4x_1+4x_2)}{32} + \frac{\cos(2\theta_1+2\theta_3-4x_1-4x_2)}{32} + \frac{\cos(2\theta_1+2\theta_3-4x_1+4x_2)}{32} - \frac{\cos(2\theta_1+2\theta_3-2x_1-4x_2)}{16} - \frac{\cos(2\theta_1+2\theta_3-2x_1+4x_2)}{16} + \frac{\cos(2\theta_1+2\theta_3+2x_1-4x_2)}{16}
\end{aligned}
$$

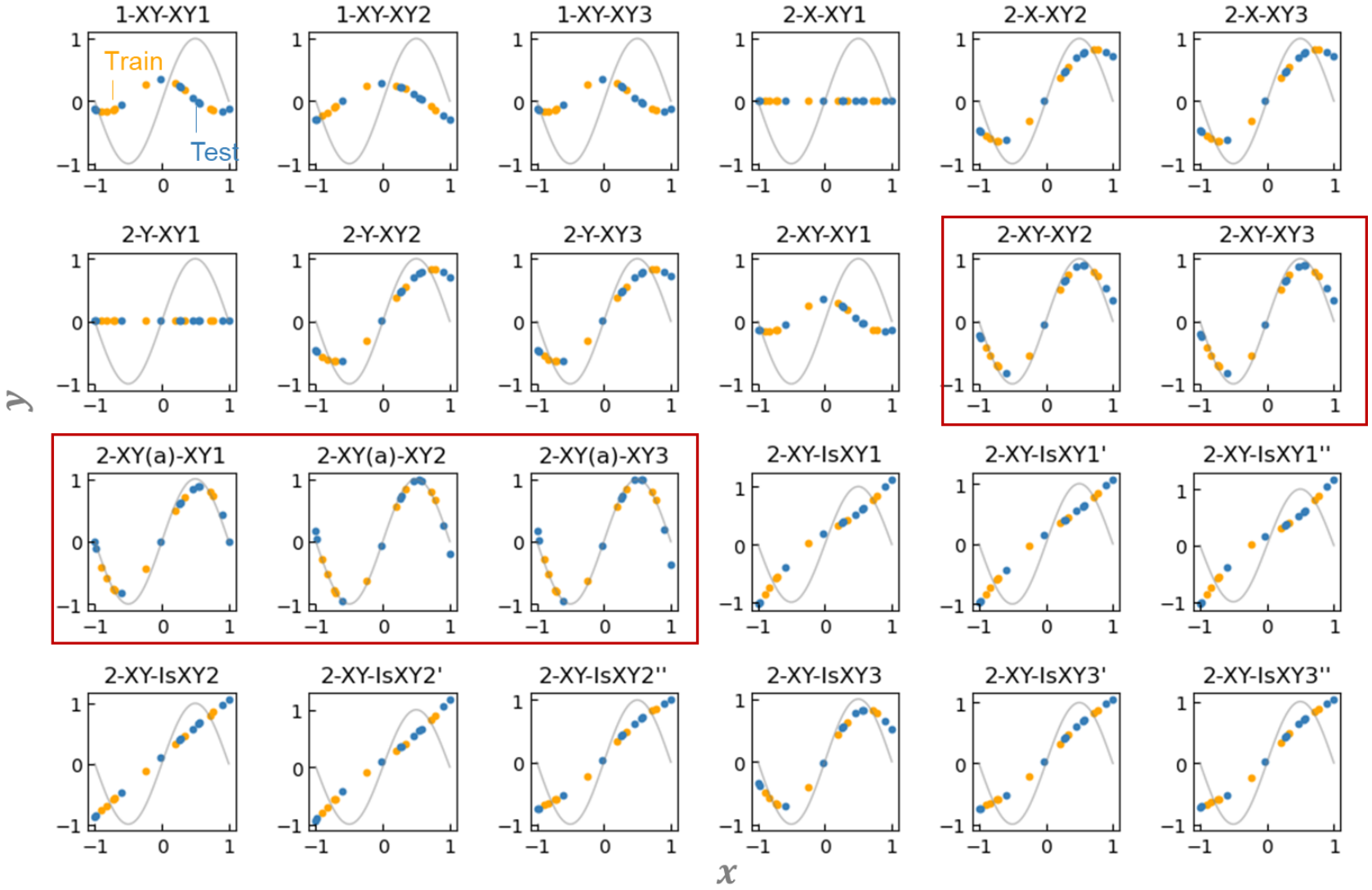
$$+\frac{\cos\left(2\theta_1+2\theta_3+2x_1+4x_2\right)}{16}+\frac{\cos\left(2\theta_1+2\theta_3+4x_1-4x_2\right)}{32}+\frac{\cos\left(2\theta_1+2\theta_3+4x_1+4x_2\right)}{32}-\frac{\cos\left(-2\theta_1-2\theta_2+2\theta_3+2x_1+4x_2\right)}{32}+\frac{\cos\left(-2\theta_1-2\theta_2+2\theta_3+4x_1+4x_2\right)}{128}-\frac{\cos\left(-2\theta_1+2\theta_2-2\theta_3+2x_1+4x_2\right)}{32}$$
$$+\frac{\cos\left(-2\theta_1+2\theta_2-2\theta_3+4x_1+4x_2\right)}{128}+\frac{\cos\left(-2\theta_1+2\theta_2+2\theta_3-4x_1+4x_2\right)}{128}+\frac{\cos\left(-2\theta_1+2\theta_2+2\theta_3-2x_1+4x_2\right)}{32}-\frac{\cos\left(-2\theta_1+2\theta_2+2\theta_3+2x_1-4x_2\right)}{32}-\frac{\cos\left(-2\theta_1+2\theta_2+2\theta_3+2x_1+4x_2\right)}{32}$$
$$+\frac{\cos\left(-2\theta_1+2\theta_2+2\theta_3+4x_1-4x_2\right)}{128}+\frac{\cos\left(-2\theta_1+2\theta_2+2\theta_3+4x_1+4x_2\right)}{128}+\frac{\cos\left(2\theta_1-2\theta_2-2\theta_3+2x_1+4x_2\right)}{32}+\frac{\cos\left(2\theta_1-2\theta_2-2\theta_3+4x_1+4x_2\right)}{128}+\frac{\cos\left(2\theta_1-2\theta_2+2\theta_3-4x_1+4x_2\right)}{128}$$
$$-\frac{\cos\left(2\theta_1-2\theta_2+2\theta_3-2x_1+4x_2\right)}{32}+\frac{\cos\left(2\theta_1-2\theta_2+2\theta_3+2x_1-4x_2\right)}{32}+\frac{\cos\left(2\theta_1-2\theta_2+2\theta_3+2x_1+4x_2\right)}{32}+\frac{\cos\left(2\theta_1-2\theta_2+2\theta_3+4x_1-4x_2\right)}{128}+\frac{\cos\left(2\theta_1-2\theta_2+2\theta_3+4x_1+4x_2\right)}{128}$$
$$+\frac{\cos\left(2\theta_1+2\theta_2-2\theta_3-4x_1+4x_2\right)}{128}-\frac{\cos\left(2\theta_1+2\theta_2-2\theta_3-2x_1+4x_2\right)}{32}+\frac{\cos\left(2\theta_1+2\theta_2-2\theta_3+2x_1-4x_2\right)}{32}+\frac{\cos\left(2\theta_1+2\theta_2-2\theta_3+2x_1+4x_2\right)}{32}+\frac{\cos\left(2\theta_1+2\theta_2-2\theta_3+4x_1-4x_2\right)}{128}$$
$$+\frac{\cos\left(2\theta_1+2\theta_2-2\theta_3+4x_1+4x_2\right)}{128}+\frac{\cos\left(2\theta_1+2\theta_2+2\theta_3-4x_1-4x_2\right)}{128}+\frac{\cos\left(2\theta_1+2\theta_2+2\theta_3-4x_1+4x_2\right)}{128}-\frac{\cos\left(2\theta_1+2\theta_2+2\theta_3-2x_1-4x_2\right)}{32}-\frac{\cos\left(2\theta_1+2\theta_2+2\theta_3-2x_1+4x_2\right)}{32}$$
$$+\frac{\cos\left(2\theta_1+2\theta_2+2\theta_3+2x_1-4x_2\right)}{32}+\frac{\cos\left(2\theta_1+2\theta_2+2\theta_3+2x_1+4x_2\right)}{32}+\frac{\cos\left(2\theta_1+2\theta_2+2\theta_3+4x_1-4x_2\right)}{128}+\frac{\cos\left(2\theta_1+2\theta_2+2\theta_3+4x_1+4x_2\right)}{128}+\frac{1}{4}$$

Eq S3

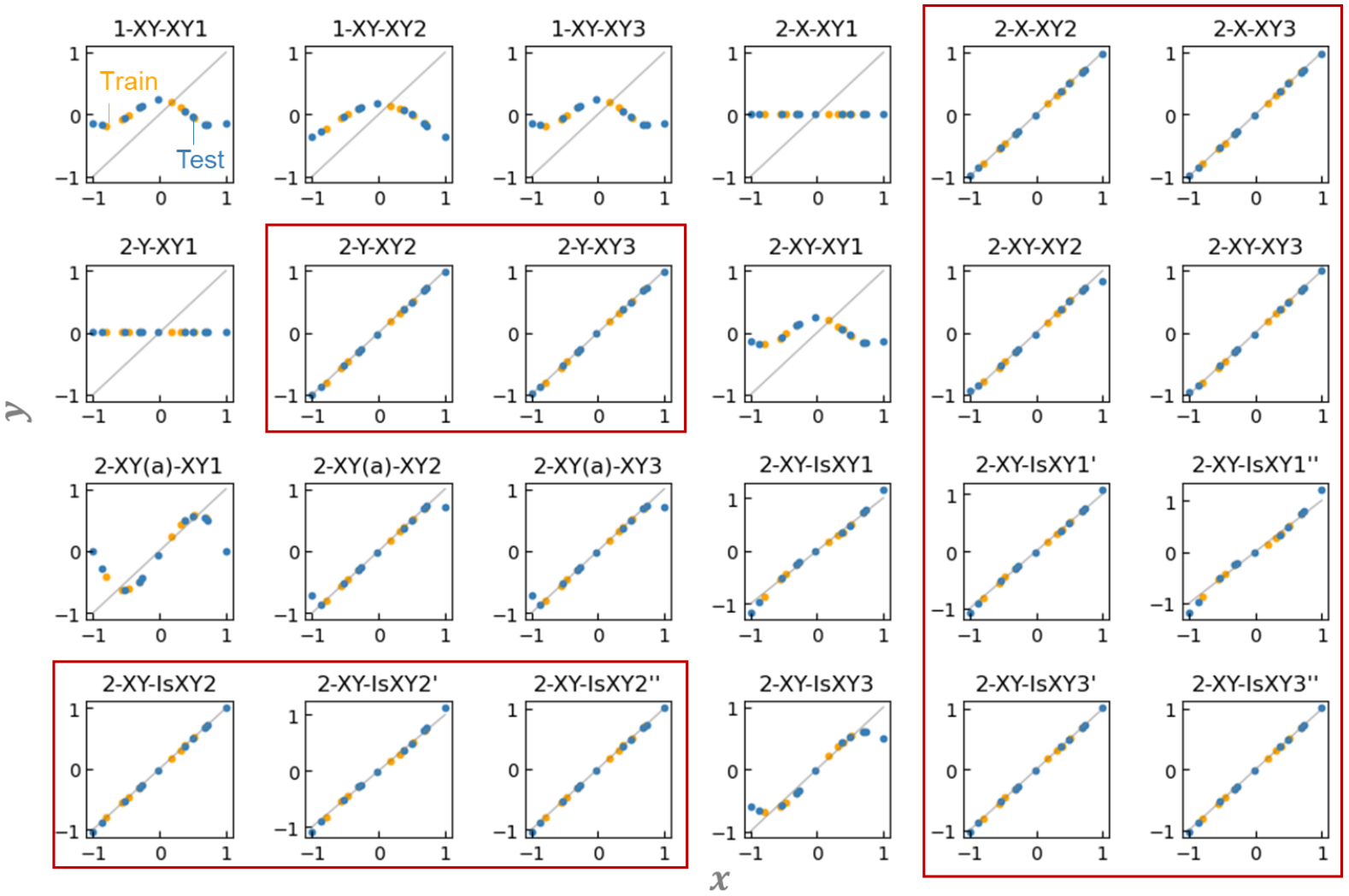
$$2\left(\sin\left(\theta_{11y}\right)\sin\left(x_1\right)\sin\left(x_2\right)+\cos\left(\theta_{11y}\right)\cos\left(x_1\right)\cos\left(x_2\right)\right)^2\cos^2\left(x_3\right)+2\left(\sin\left(\theta_{11y}\right)\sin\left(x_1\right)\cos\left(x_2\right)-\sin\left(x_2\right)\cos\left(\theta_{11y}\right)\cos\left(x_1\right)\right)^2\sin^2\left(x_3\right)-1$$

Eq S4

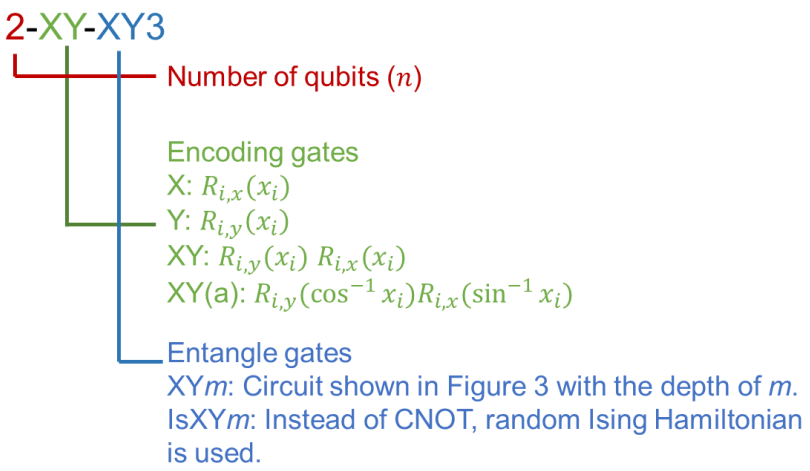
$$-4\sin^2\left(\theta_{11y}\right)\sin^4\left(x_1\right)+4\sin^2\left(\theta_{11y}\right)\sin^2\left(x_1\right)-16\sin^2\left(\theta_{11y}\right)\sin^4\left(x_2\right)\sin^4\left(x_3\right)+16\sin^2\left(\theta_{11y}\right)\sin^4\left(x_2\right)\sin^2\left(x_3\right)-4\sin^2\left(\theta_{11y}\right)\sin^4\left(x_2\right)+16\sin^2\left(\theta_{11y}\right)\sin^2\left(x_2\right)\sin^4\left(x_3\right)-16\sin^2\left(\theta_{11y}\right)\sin^2\left(x_2\right)\sin^2\left(x_3\right)+4\sin^2\left(\theta_{11y}\right)\sin^2\left(x_2\right)-4\sin^2\left(\theta_{11y}\right)\sin^4\left(x_3\right)+4\sin^2\left(\theta_{11y}\right)\sin^2\left(x_3\right)-2\sin^2\left(\theta_{11y}\right)-16\sin\left(\theta_{11y}\right)\sin^3\left(x_1\right)\sin^3\left(x_2\right)\cos\left(\theta_{11y}\right)\cos\left(x_1\right)\cos\left(x_2\right)+8\sin\left(\theta_{11y}\right)\sin^3\left(x_1\right)\sin\left(x_2\right)\cos\left(\theta_{11y}\right)\cos\left(x_1\right)\cos\left(x_2\right)+8\sin\left(\theta_{11y}\right)\sin\left(x_1\right)\sin^3\left(x_2\right)\cos\left(\theta_{11y}\right)\cos\left(x_1\right)\cos\left(x_2\right)+16\sin\left(\theta_{11y}\right)\sin\left(x_1\right)\sin\left(x_2\right)\sin^4\left(x_3\right)\cos\left(\theta_{11y}\right)\cos\left(x_1\right)\cos\left(x_2\right)-16\sin\left(\theta_{11y}\right)\sin\left(x_1\right)\sin\left(x_2\right)\sin^2\left(x_3\right)\cos\left(\theta_{11y}\right)\cos\left(x_1\right)\cos\left(x_2\right)+32\sin^4\left(x_1\right)\sin^4\left(x_2\right)\sin^4\left(x_3\right)-32\sin^4\left(x_1\right)\sin^4\left(x_2\right)\sin^2\left(x_3\right)+8\sin^4\left(x_1\right)\sin^4\left(x_2\right)-32\sin^4\left(x_1\right)\sin^2\left(x_2\right)\sin^4\left(x_3\right)+32\sin^4\left(x_1\right)\sin^2\left(x_2\right)\sin^2\left(x_3\right)-8\sin^4\left(x_1\right)\sin^2\left(x_2\right)+8\sin^4\left(x_1\right)\sin^4\left(x_3\right)-8\sin^4\left(x_1\right)\sin^2\left(x_3\right)+4\sin^4\left(x_1\right)-32\sin^2\left(x_1\right)\sin^4\left(x_2\right)\sin^4\left(x_3\right)+32\sin^2\left(x_1\right)\sin^4\left(x_2\right)\sin^2\left(x_3\right)-8\sin^2\left(x_1\right)\sin^4\left(x_2\right)+32\sin^2\left(x_1\right)\sin^2\left(x_2\right)\sin^4\left(x_3\right)-32\sin^2\left(x_1\right)\sin^2\left(x_2\right)\sin^2\left(x_3\right)+8\sin^2\left(x_1\right)\sin^2\left(x_2\right)-8\sin^2\left(x_1\right)\sin^4\left(x_3\right)+8\sin^2\left(x_1\right)\sin^2\left(x_3\right)-4\sin^2\left(x_1\right)+16\sin^4\left(x_2\right)\sin^4\left(x_3\right)-16\sin^4\left(x_2\right)\sin^2\left(x_3\right)+4\sin^4\left(x_2\right)-16\sin^2\left(x_2\right)\sin^4\left(x_3\right)+16\sin^2\left(x_2\right)\sin^2\left(x_3\right)-4\sin^2\left(x_2\right)+4\sin^4\left(x_3\right)-4\sin^2\left(x_3\right)+1$$


1-XY-XY1
Train
Test
1-XY-XY2
1-XY-XY3
2-X-XY1
2-X-XY2
2-X-XY3
2-Y-XY1
2-Y-XY2
2-Y-XY3
2-XY-XY1
2-XY-XY2
2-XY-XY3
2-XY(a)-XY1
2-XY(a)-XY2
2-XY(a)-XY3
2-XY-IsXY1
2-XY-IsXY1'
2-XY-IsXY1''
2-XY-IsXY2
2-XY-IsXY2'
2-XY-IsXY2''
2-XY-IsXY3
2-XY-IsXY3'
2-XY-IsXY3''
y
x

a)


1-XY-XY1
Train
Test
1-XY-XY2
1-XY-XY3
2-X-XY1
2-X-XY2
2-X-XY3
2-Y-XY1
2-Y-XY2
2-Y-XY3
2-XY-XY1
2-XY-XY2
2-XY-XY3
2-XY(a)-XY1
2-XY(a)-XY2
2-XY(a)-XY3
2-XY-IsXY1
2-XY-IsXY1'
2-XY-IsXY1''
2-XY-IsXY2
2-XY-IsXY2'
2-XY-IsXY2''
2-XY-IsXY3
2-XY-IsXY3'
2-XY-IsXY3''
y
x

b)



c)

Figure S 1 Regression results with different quantum circuits, fitting a) $y = \sin(x)$ and b) $y = x$. Successful results are marked red. Regressions were repeated three times with Ising-type circuits (IsXYm) because the results changed randomly. c) Explanation of circuit configuration.

NOTE: The best circuit configuration (2-XY-XYm) was selected for the following reasons. One qubit circuit could not fit the one-dimensional functions (e.g., 1-XY-XY3). For initial encoding, only the use of R_y gates was sufficient to fit the linear function (e.g., 2-Y-XY2 and 2-Y-XY3). However, additional R_x gates were needed for a non-linear $\sin(x)$ function (e.g., 2-XY-XY2 and 2-XY-XY3). Preprocessing of explanatory variables with $\cos^{-1} x_i$ and $\sin^{-1} x_i$ was not successful with the linear function (e.g., 2-XY(a)-XY3). The use of Ising Hamiltonian instead of CNOT circuits led to more unstable regressions due to the randomness (e.g., 2-XY-IsXY3).

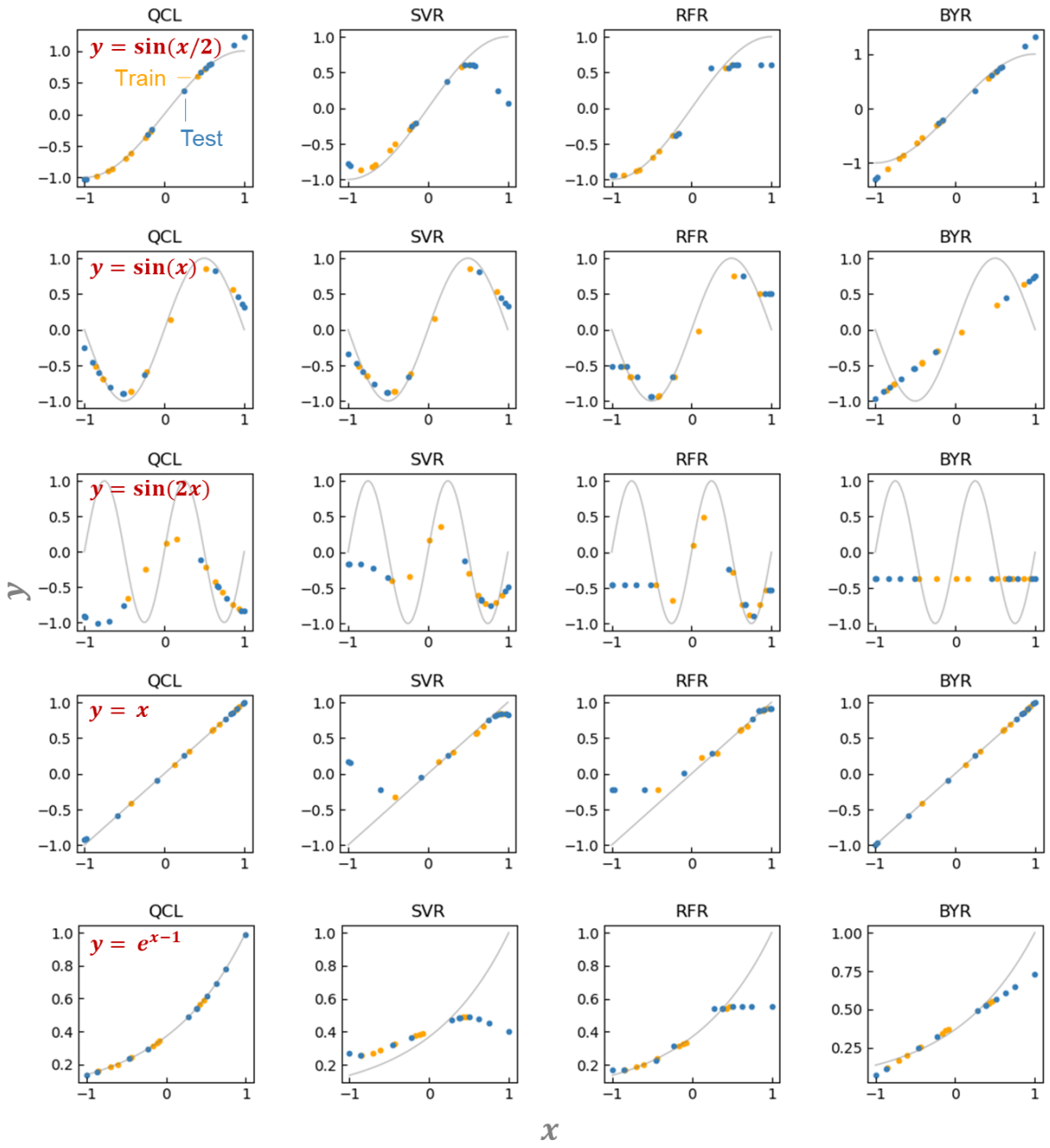
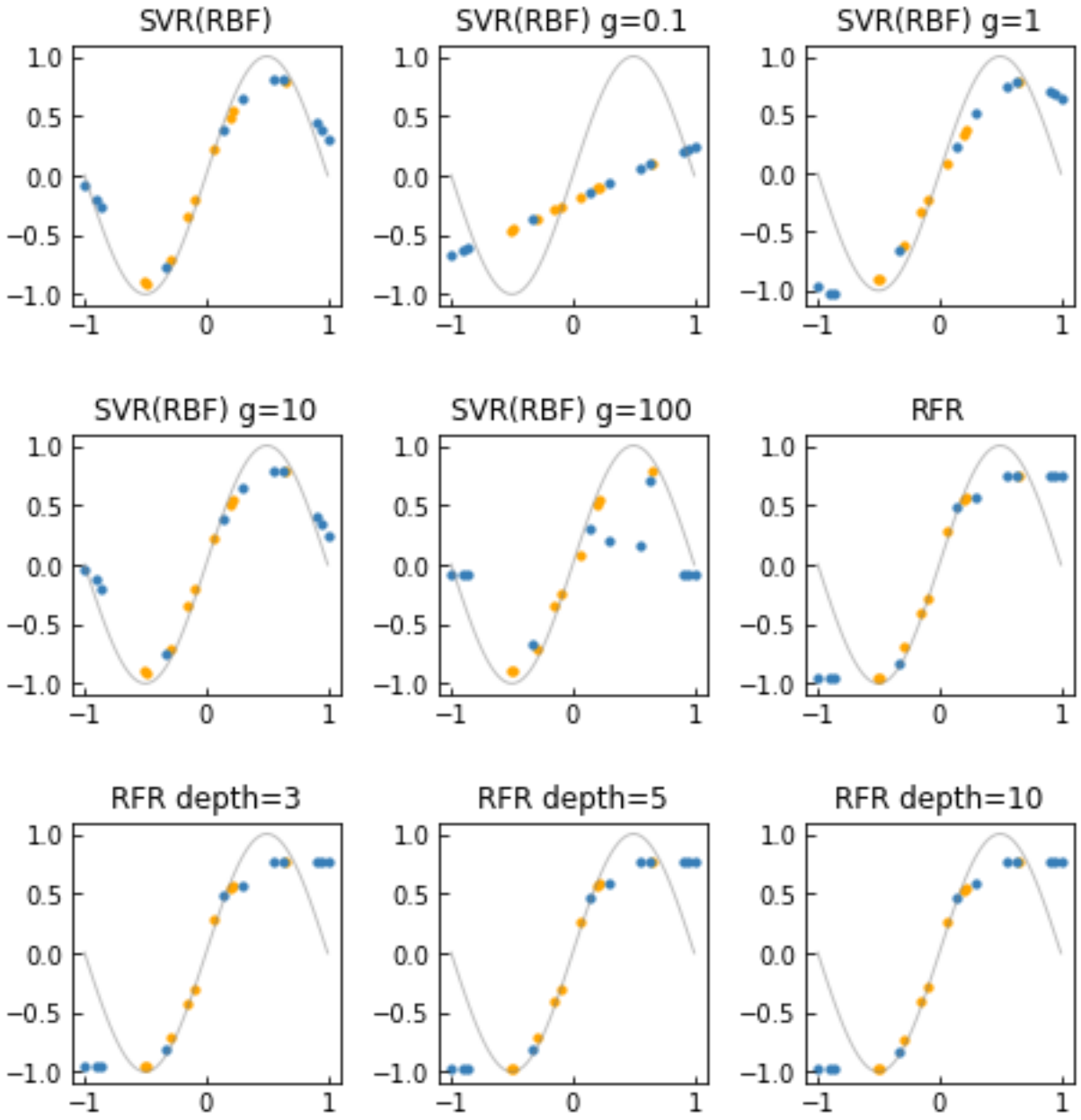
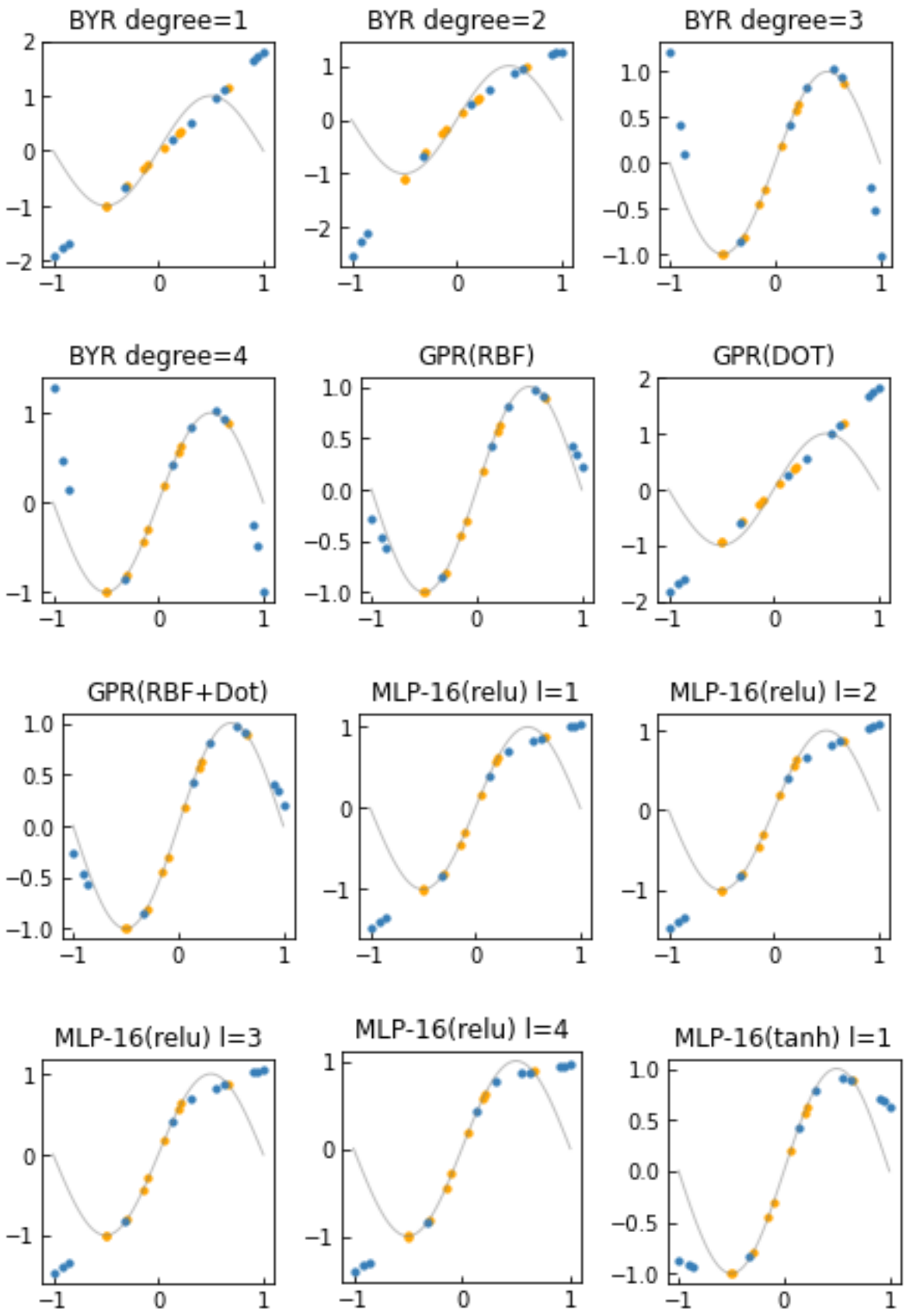
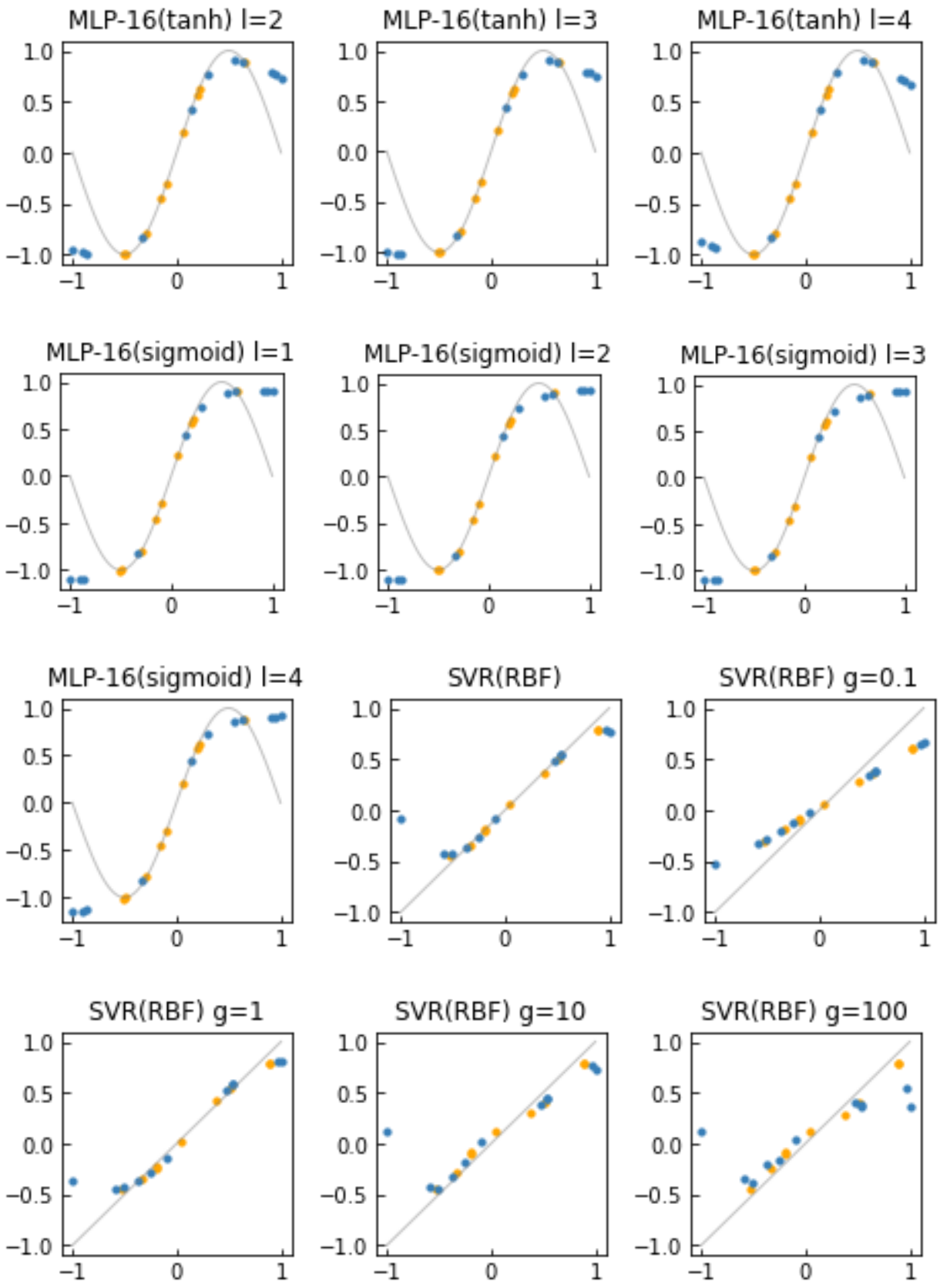
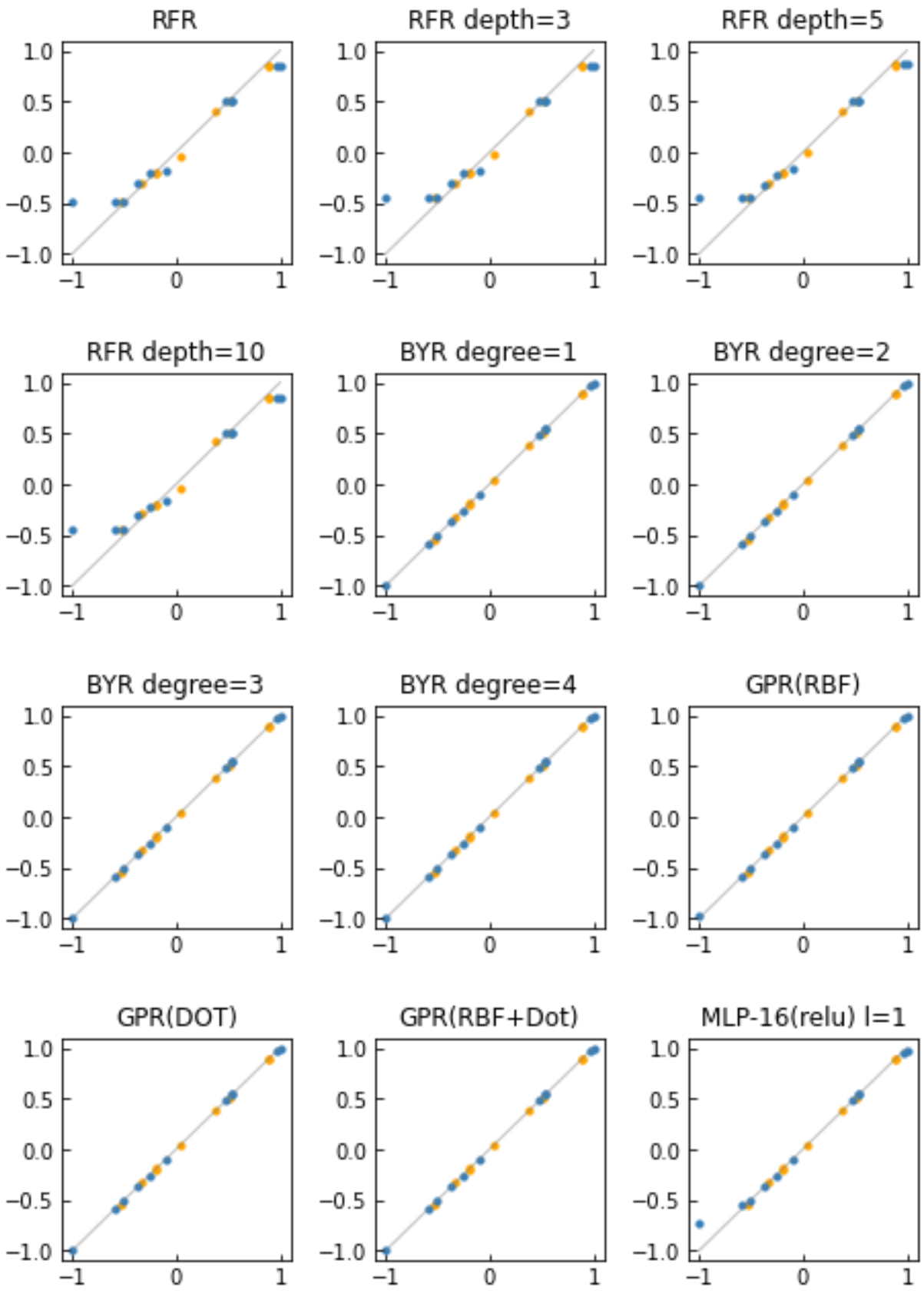


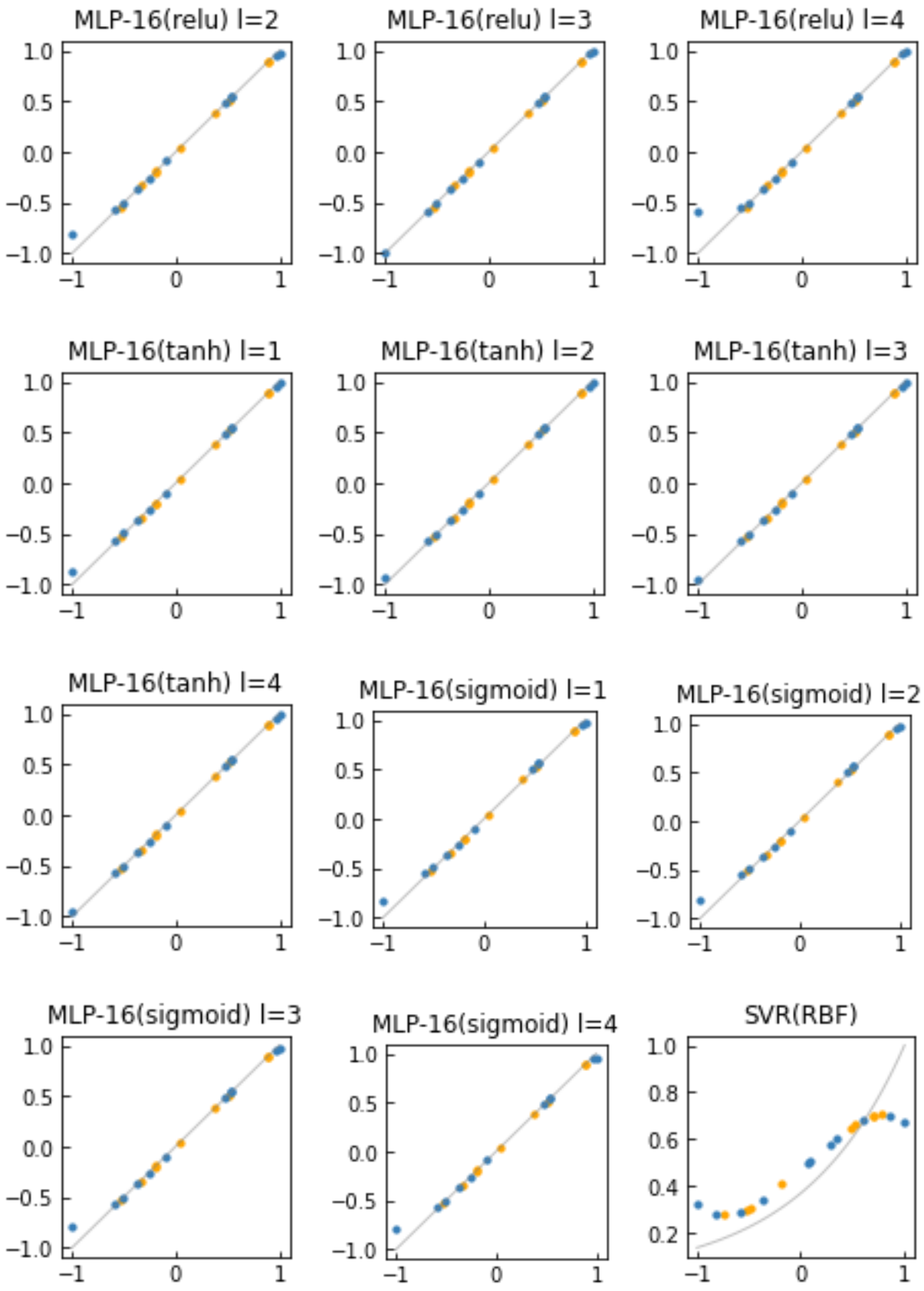
Figure S 2 Full regression results for Figure 4a.

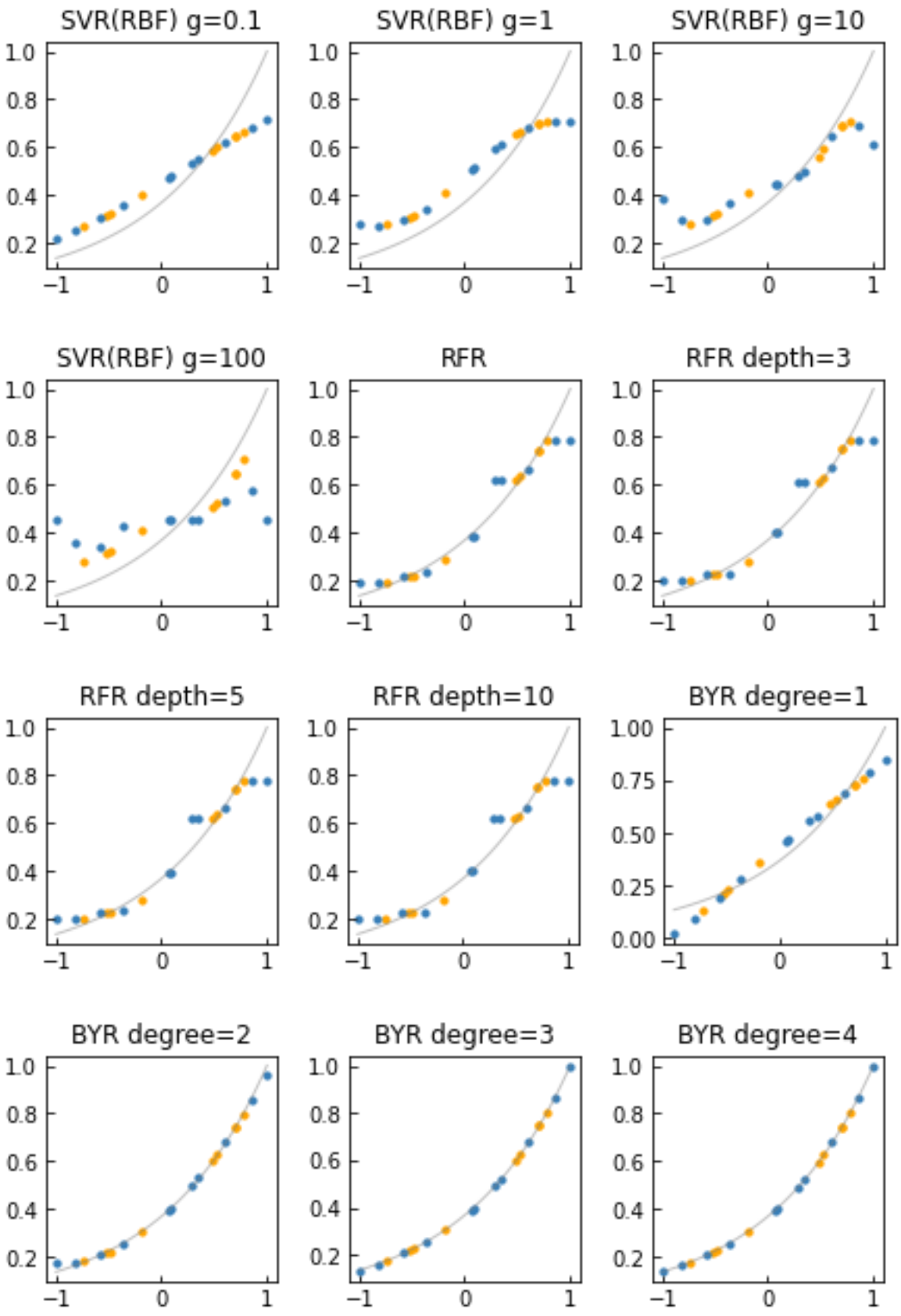

SVR(RBF)
SVR(RBF) g=0.1
SVR(RBF) g=1
SVR(RBF) g=10
SVR(RBF) g=100
RFR
RFR depth=3
RFR depth=5
RFR depth=10

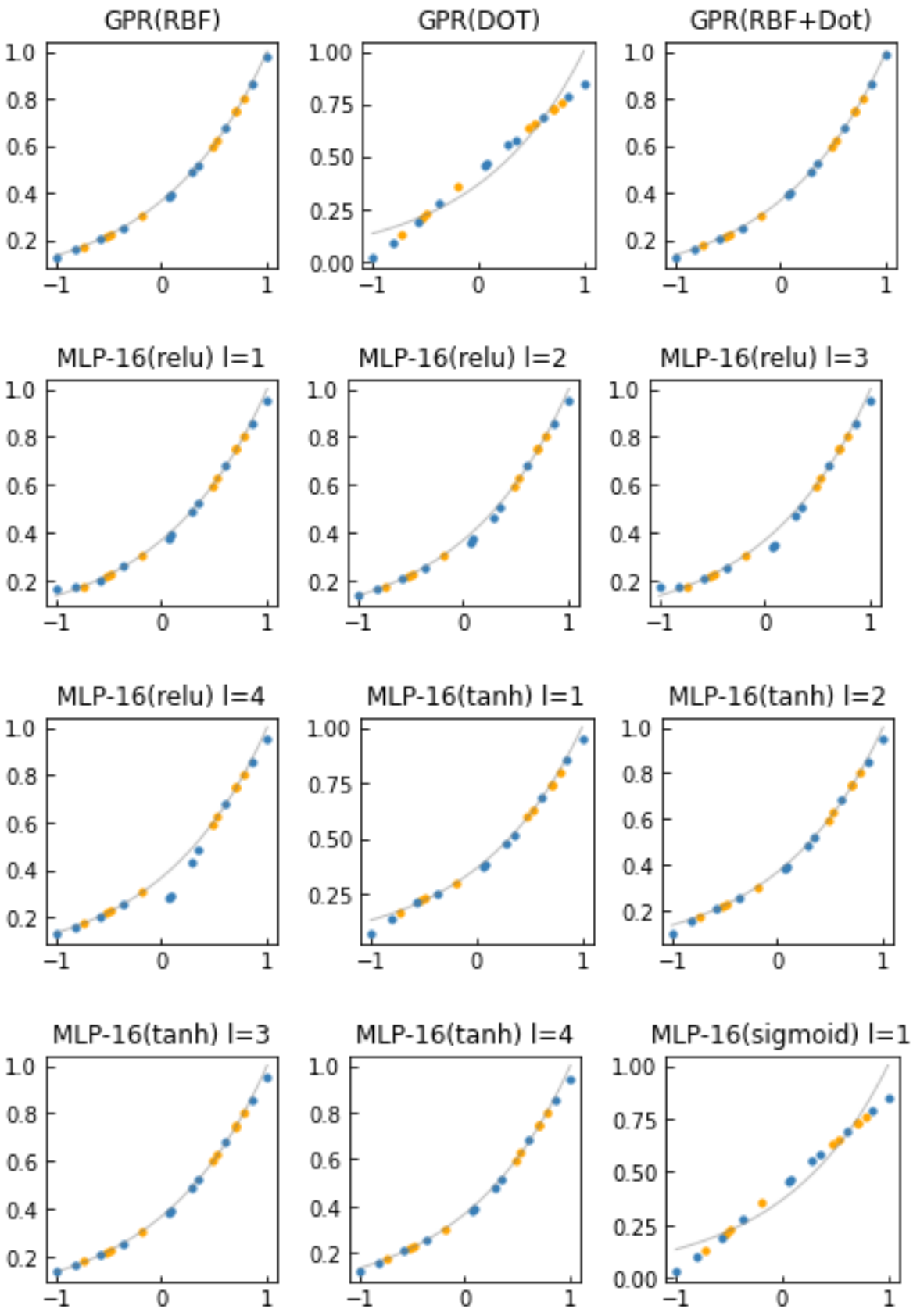

BYR degree=1
BYR degree=2
BYR degree=3
BYR degree=4
GPR(RBF)
GPR(DOT)
GPR(RBF+Dot)
MLP-16(relu) l=1
MLP-16(relu) l=2
MLP-16(relu) l=3
MLP-16(relu) l=4
MLP-16(tanh) l=1


MLP-16(tanh) l=2
MLP-16(tanh) l=3
MLP-16(tanh) l=4
MLP-16(sigmoid) l=1
MLP-16(sigmoid) l=2
MLP-16(sigmoid) l=3
MLP-16(sigmoid) l=4
SVR(RBF)
SVR(RBF) g=0.1
SVR(RBF) g=1
SVR(RBF) g=10
SVR(RBF) g=100


RFR
RFR depth=3
RFR depth=5
RFR depth=10
BYR degree=1
BYR degree=2
BYR degree=3
BYR degree=4
GPR(RBF)
GPR(DOT)
GPR(RBF+Dot)
MLP-16(relu) l=1


MLP-16(relu) l=2
MLP-16(relu) l=3
MLP-16(relu) l=4
MLP-16(tanh) l=1
MLP-16(tanh) l=2
MLP-16(tanh) l=3
MLP-16(tanh) l=4
MLP-16(sigmoid) l=1
MLP-16(sigmoid) l=2
MLP-16(sigmoid) l=3
MLP-16(sigmoid) l=4
SVR(RBF)


SVR(RBF) g=0.1
SVR(RBF) g=1
SVR(RBF) g=10
SVR(RBF) g=100
RFR
RFR depth=3
RFR depth=5
RFR depth=10
BYR degree=1
BYR degree=2
BYR degree=3
BYR degree=4


GPR(RBF)
GPR(DOT)
GPR(RBF+Dot)
MLP-16(relu) l=1
MLP-16(relu) l=2
MLP-16(relu) l=3
MLP-16(relu) l=4
MLP-16(tanh) l=1
MLP-16(tanh) l=2
MLP-16(tanh) l=3
MLP-16(tanh) l=4
MLP-16(sigmoid) l=1

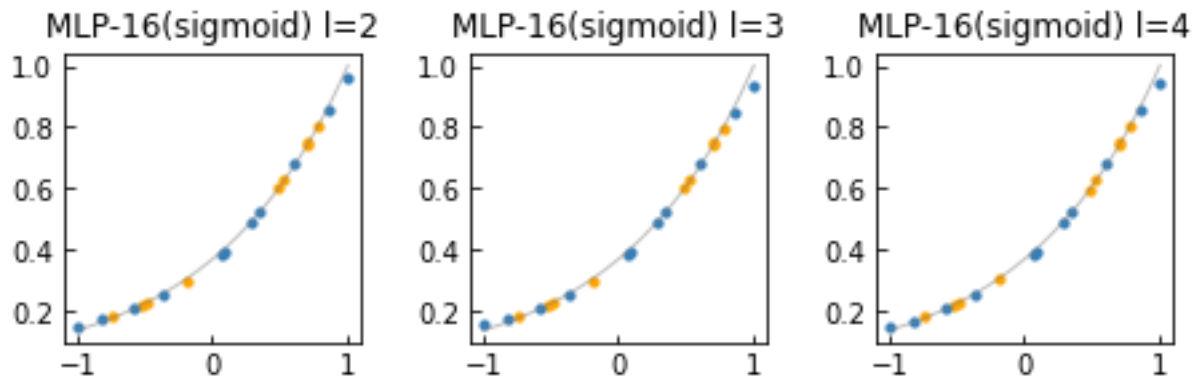
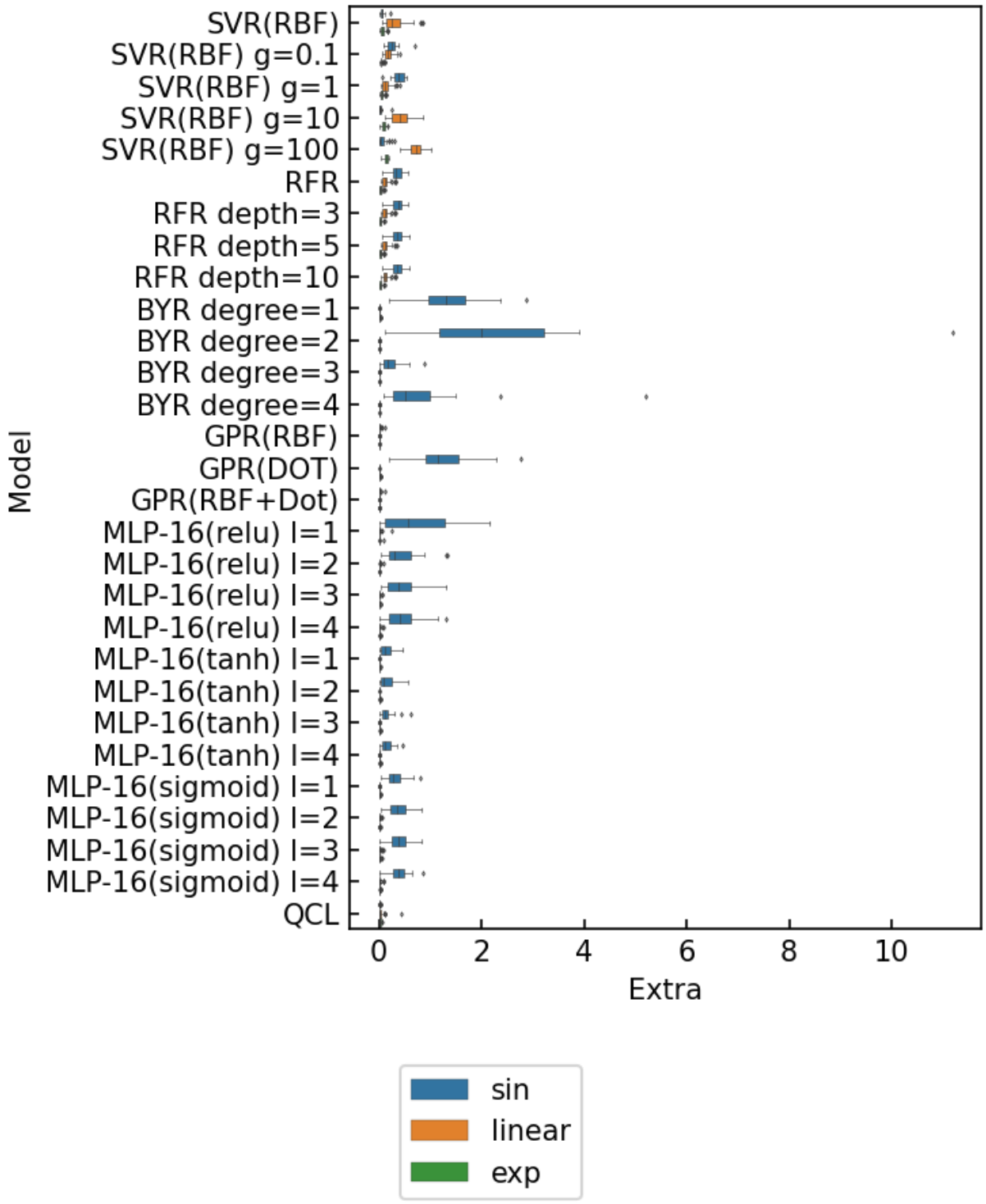


Figure S 3 Additional regression results for Figure S 2 using conventional machine learning models with various hyperparameters. Details of the models are explained in Table S2. The polynomial regression by BYR (degree > 1) could basically fit the three functions. However, the regression was unstable; it could easily induce overfitting and substantial prediction errors, as observed in Figure S 4 and Table S3.

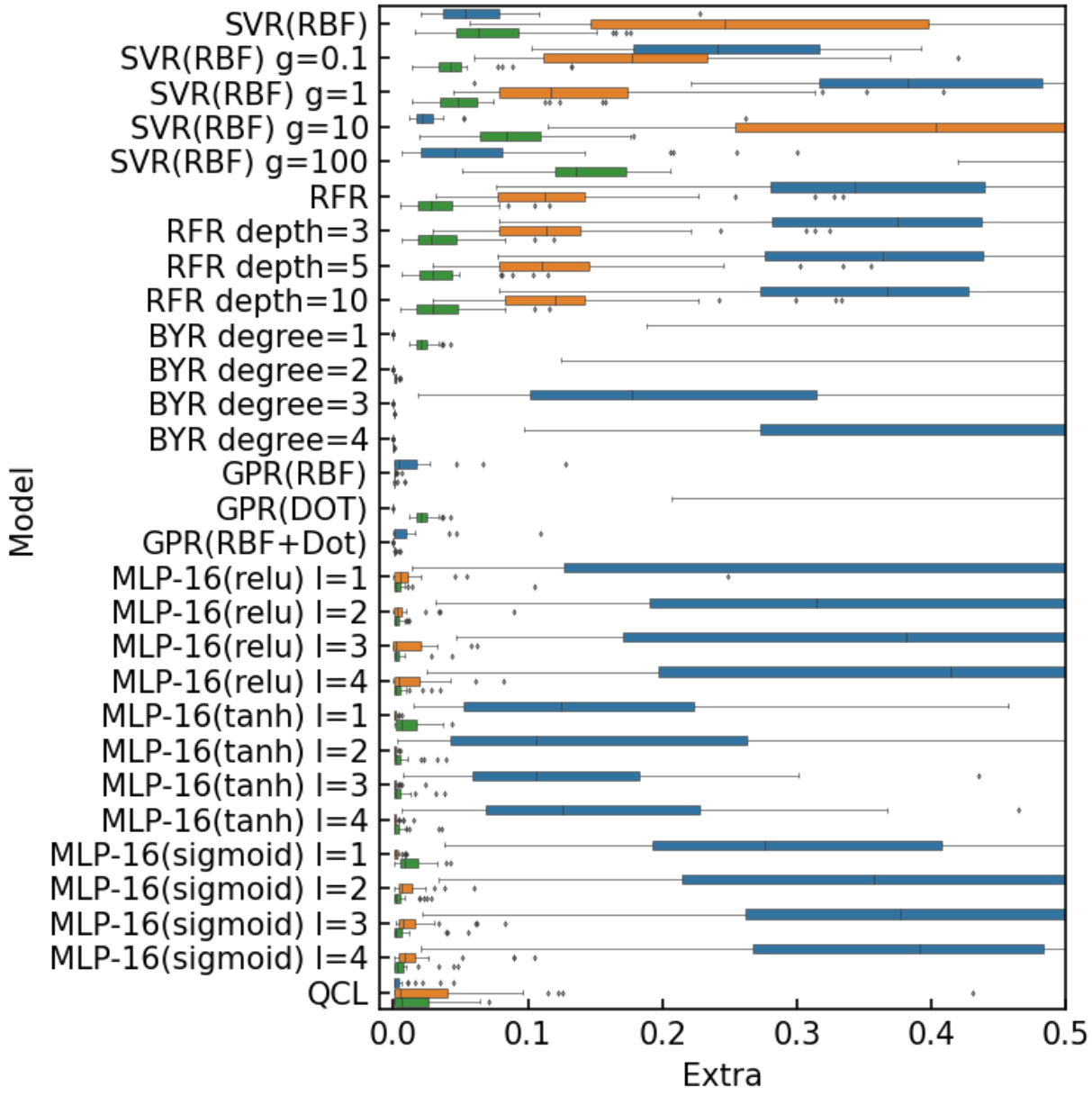
Table S2 Explanations of the conventional models.[a)]

Expression	Model name	Hyperparameter
SVR(RBF)	SVR	kernel = "rbf", gamma = "auto"
SVR(RBF) g=0.1	SVR	kernel = "rbf", gamma = 0.1
SVR(RBF) g=1	SVR	kernel = "rbf" , gamma = 1
SVR(RBF) g=10	SVR	kernel = "rbf" , gamma = 10
SVR(RBF) g=100	SVR	kernel = "rbf" , gamma = 100
RFR	RandomForestRegressor	(default)
RFR depth=3	RandomForestRegressor	max_depth=3
RFR depth=5	RandomForestRegressor	max_depth=5
RFR depth=10	RandomForestRegressor	max_depth=10
BYR degree=1	BayesianRidge	(default)
BYR degree=2	BayesianRidge	(default) Convert x to $x + x^2$
BYR degree=3	BayesianRidge	(default) Convert x to $x + x^2 + x^3$
BYR degree=4	BayesianRidge	(default) Convert x to $x + x^2 \ldots + x^4$
GPR(RBF)	GaussianProcessRegressor	kernel=RBF + WhiteKernel
GPR(DOT)	GaussianProcessRegressor	kernel=DotProduct + WhiteKernel
GPR(RBF+Dot)	GaussianProcessRegressor	kernel=RBF+DotProduct + WhiteKernel
MLP-16(relu) l=1	Multi layer perceptron[b)]	One hidden layer, ReLu activation
MLP-16(relu) l=2	Multi layer perceptron[b)]	Two hidden layers, ReLu activation
MLP-16(relu) l=3	Multi layer perceptron[b)]	Three hidden layers, ReLu activation
MLP-16(relu) l=4	Multi layer perceptron[b)]	Four hidden layers, ReLu activation
MLP-16(tanh) l=1	Multi layer perceptron[b)]	One hidden layer, tanh activation
MLP-16(tanh) l=2	Multi layer perceptron[b)]	Two hidden layers, tanh activation
MLP-16(tanh) l=3	Multi layer perceptron[b)]	Three hidden layers, tanh activation
MLP-16(tanh) l=4	Multi layer perceptron[b)]	Four hidden layers, tanh activation
MLP-16(sigmoid) l=1	Multi layer perceptron[b)]	One hidden layer, sigmoid activation
MLP-16(sigmoid) l=2	Multi layer perceptron[b)]	Two hidden layers, sigmoid activation
MLP-16(sigmoid) l=3	Multi layer perceptron[b)]	Three hidden layers, sigmoid activation
MLP-16(sigmoid) l=4	Multi layer perceptron[b)]	Four hidden layers, sigmoid activation

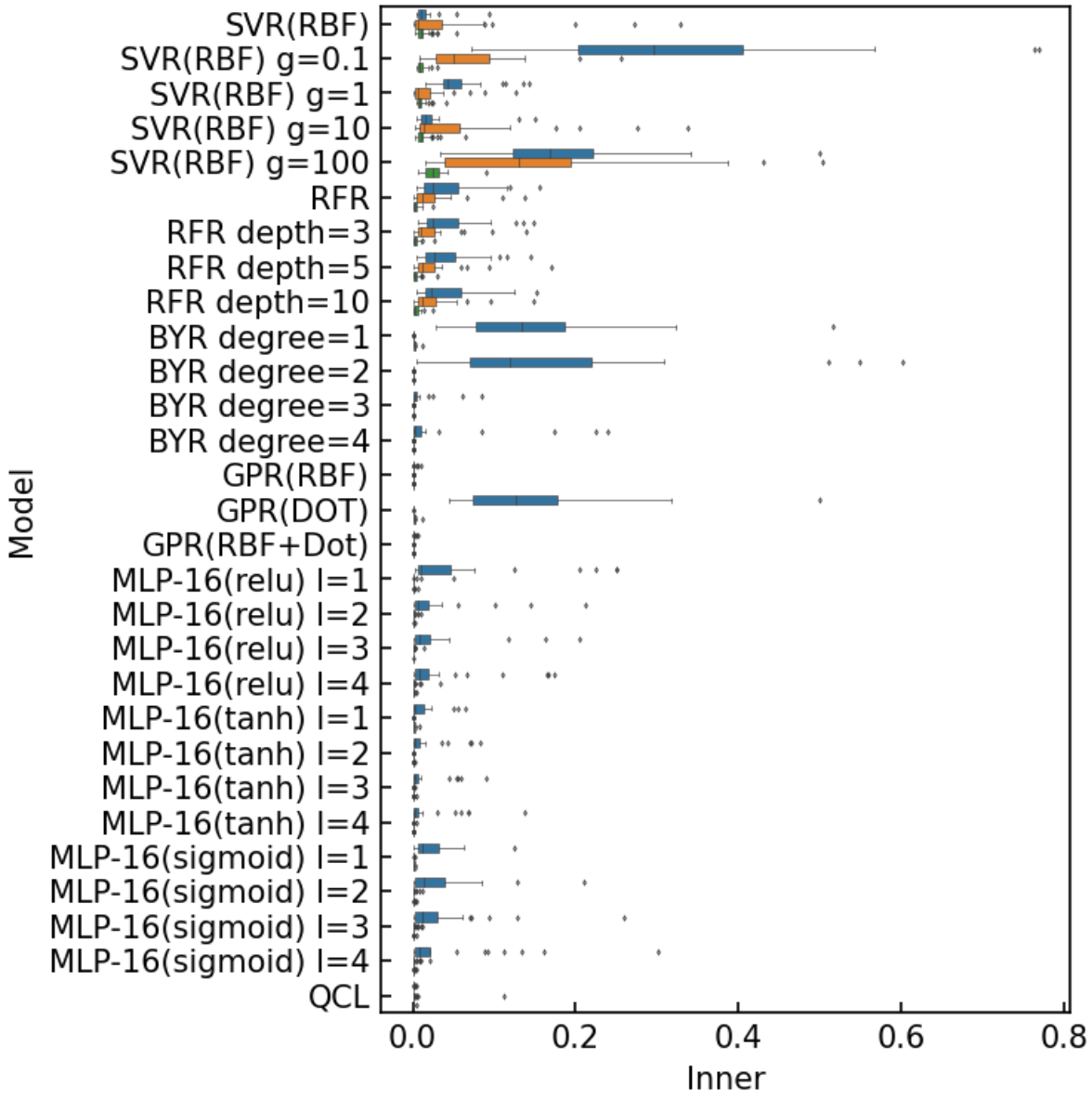
a) Except for MLP, regressions models were made using a scikit-learn (version 1.0.2) library. Default hyperparameters were used unless noted otherwise. The document is available at https://scikit-learn.org/stable/whats_new/v1.0.html.
b) MLP was implemented by a Keras (version 2.9.0) library. The dimension of the hidden layers was 16.


SVR(RBF)
SVR(RBF) g=0.1
SVR(RBF) g=1
SVR(RBF) g=10
SVR(RBF) g=100
RFR
RFR depth=3
RFR depth=5
RFR depth=10
BYR degree=1
BYR degree=2
BYR degree=3
BYR degree=4
GPR(RBF)
GPR(DOT)
GPR(RBF+Dot)
MLP-16(relu) l=1
MLP-16(relu) l=2
MLP-16(relu) l=3
MLP-16(relu) l=4
MLP-16(tanh) l=1
MLP-16(tanh) l=2
MLP-16(tanh) l=3
MLP-16(tanh) l=4
MLP-16(sigmoid) l=1
MLP-16(sigmoid) l=2
MLP-16(sigmoid) l=3
MLP-16(sigmoid) l=4
QCL
Model
0
2
4
6
8
10
Extra
sin
linear
exp

a)


SVR(RBF)
SVR(RBF) g=0.1
SVR(RBF) g=1
SVR(RBF) g=10
SVR(RBF) g=100
RFR
RFR depth=3
RFR depth=5
RFR depth=10
BYR degree=1
BYR degree=2
BYR degree=3
BYR degree=4
GPR(RBF)
GPR(DOT)
GPR(RBF+Dot)
MLP-16(relu) l=1
MLP-16(relu) l=2
MLP-16(relu) l=3
MLP-16(relu) l=4
MLP-16(tanh) l=1
MLP-16(tanh) l=2
MLP-16(tanh) l=3
MLP-16(tanh) l=4
MLP-16(sigmoid) l=1
MLP-16(sigmoid) l=2
MLP-16(sigmoid) l=3
MLP-16(sigmoid) l=4
QCL
Model
0.0
0.1
0.2
0.3
0.4
0.5
Extra

b)


SVR(RBF)
SVR(RBF) g=0.1
SVR(RBF) g=1
SVR(RBF) g=10
SVR(RBF) g=100
RFR
RFR depth=3
RFR depth=5
RFR depth=10
BYR degree=1
BYR degree=2
BYR degree=3
BYR degree=4
GPR(RBF)
GPR(DOT)
GPR(RBF+Dot)
MLP-16(relu) l=1
MLP-16(relu) l=2
MLP-16(relu) l=3
MLP-16(relu) l=4
MLP-16(tanh) l=1
MLP-16(tanh) l=2
MLP-16(tanh) l=3
MLP-16(tanh) l=4
MLP-16(sigmoid) l=1
MLP-16(sigmoid) l=2
MLP-16(sigmoid) l=3
MLP-16(sigmoid) l=4
QCL
Model
0.0
0.2
0.4
0.6
0.8
Inner

c)

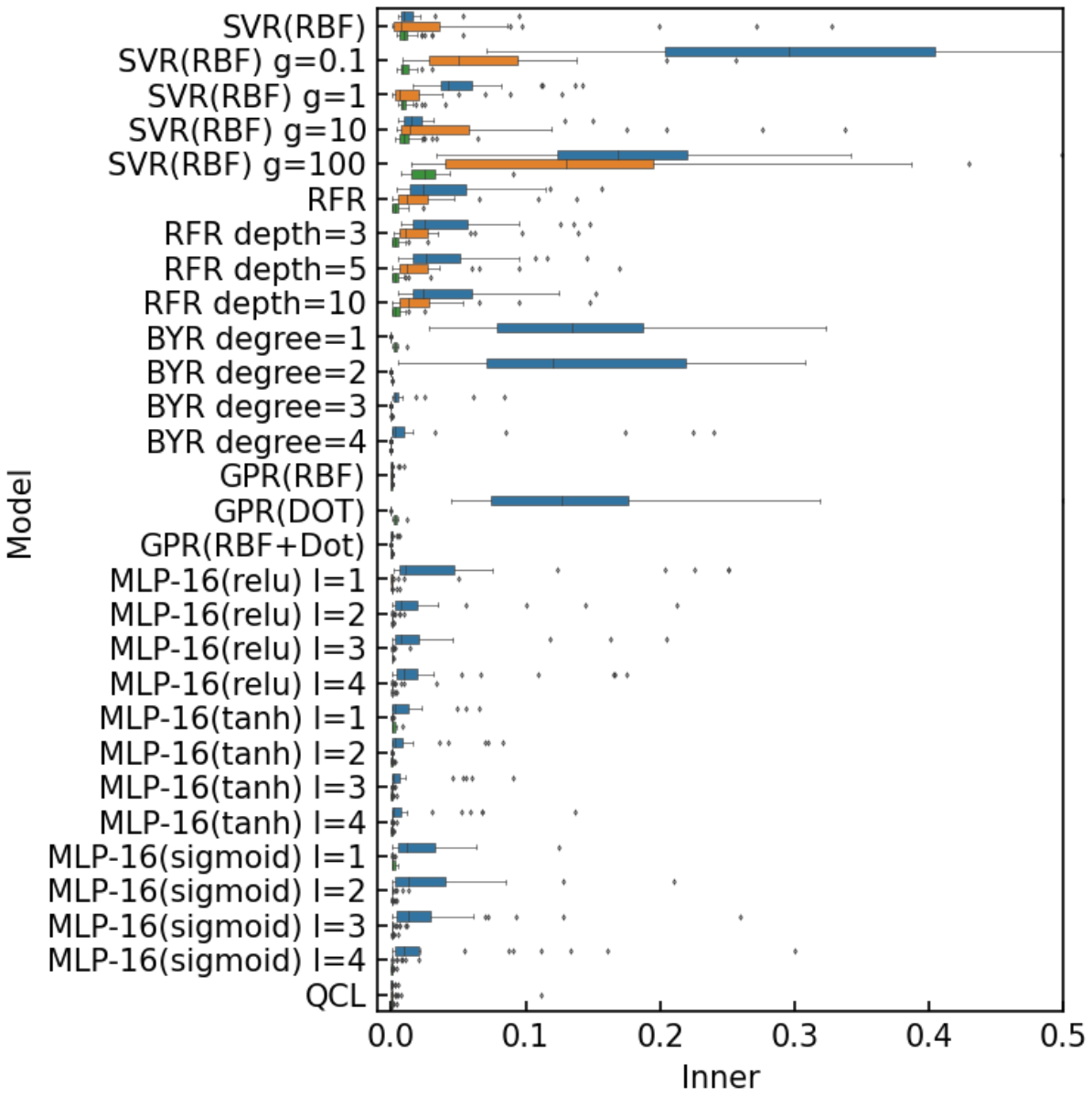


Figure S 4 Mean squared errors (MSEs) for the one-dimensional regression tasks. The random data preparation and regressions were repeated 30 times. a,b) Prediction errors for the testing datasets in the extrapolating regions and c,d) for interpolating regions. For clearer comparison, enlarged graphs are shown in Figures b and d) by setting the *x*-range of 0 to 0.5.

Table S3 Average MSEs for the regression task in Figure S 4. Extra and inner represent the testing data in the extrapolating and interpolating regions, respectively. The "Total" column is the sum of Extra (all) and Inner (all), which are average MSEs for the regression tasks of linear, sin, and exponential curves.

Model	Extra (all)	Inner (all)	Total	Extra (linear)	Inner (linear)	Extra (sin)	Inner (sin)	Extra (exp)	Inner (exp)
GPR(RBF+Dot)	0.0034	0.00018	0.0036	3.3e-11	1.2e-11	0.0096	0.00052	0.00069	2.3e-05
GPR(RBF)	0.0053	0.00027	0.0056	0.00063	6.9e-05	0.014	0.0007	0.0011	4e-05
QCL	0.02	0.0018	0.022	0.039	0.0045	0.0052	0.00037	0.017	0.00043
MLP-16(tanh) l=3	0.049	0.0042	0.054	0.002	0.00027	0.14	0.012	0.0054	0.00029
MLP-16(tanh) l=1	0.054	0.004	0.058	0.0011	0.00012	0.15	0.011	0.011	0.0013
MLP-16(tanh) l=4	0.053	0.0054	0.059	0.002	0.00031	0.15	0.016	0.0046	0.00021
MLP-16(tanh) l=2	0.054	0.0044	0.059	0.00097	9.3e-05	0.16	0.013	0.0059	0.0003
BYR degree=3	0.08	0.0029	0.083	8.3e-11	7.5e-12	0.24	0.0086	0.0001	3.3e-06
MLP-16(sigmoid) l=1	0.11	0.0079	0.12	0.0027	0.0003	0.32	0.022	0.012	0.0013
MLP-16(sigmoid) l=2	0.13	0.011	0.14	0.011	0.0013	0.37	0.03	0.0056	0.00042
MLP-16(sigmoid) l=3	0.13	0.011	0.14	0.015	0.0017	0.37	0.032	0.0071	0.0004
MLP-16(sigmoid) l=4	0.13	0.013	0.15	0.018	0.0021	0.37	0.037	0.0075	0.00038
MLP-16(relu) l=3	0.16	0.009	0.17	0.012	0.00095	0.45	0.026	0.0048	0.00037
MLP-16(relu) l=2	0.16	0.0088	0.17	0.0084	0.001	0.46	0.025	0.0031	0.00026
MLP-16(relu) l=4	0.16	0.011	0.17	0.014	0.0021	0.45	0.031	0.0052	0.0006
SVR(RBF)	0.15	0.024	0.18	0.31	0.044	0.062	0.015	0.078	0.013
RFR	0.17	0.022	0.2	0.13	0.023	0.35	0.039	0.038	0.0043
RFR depth=3	0.18	0.023	0.2	0.13	0.023	0.36	0.041	0.038	0.0043
RFR depth=10	0.18	0.023	0.2	0.13	0.024	0.36	0.041	0.038	0.0043
RFR depth=5	0.18	0.023	0.2	0.13	0.024	0.36	0.04	0.038	0.0043
SVR(RBF) g=10	0.18	0.031	0.21	0.42	0.056	0.032	0.024	0.093	0.013
SVR(RBF) g=1	0.2	0.028	0.23	0.15	0.019	0.39	0.055	0.058	0.011
MLP-16(relu) l=1	0.25	0.018	0.27	0.017	0.0024	0.72	0.05	0.0066	0.00048
BYR degree=4	0.28	0.0093	0.29	8.2e-08	1.1e-08	0.84	0.028	1.7e-05	2.6e-07
SVR(RBF) g=0.1	0.17	0.13	0.3	0.19	0.067	0.26	0.33	0.048	0.01
SVR(RBF) g=100	0.31	0.12	0.43	0.73	0.15	0.071	0.18	0.14	0.026
GPR(DOT)	0.43	0.048	0.48	2.2e-11	7.6e-12	1.3	0.14	0.022	0.0028
BYR degree=1	0.46	0.049	0.5	2e-14	7.8e-15	1.3	0.14	0.022	0.0028
BYR degree=2	0.77	0.057	0.83	2e-12	4.3e-13	2.3	0.17	0.0016	0.00011

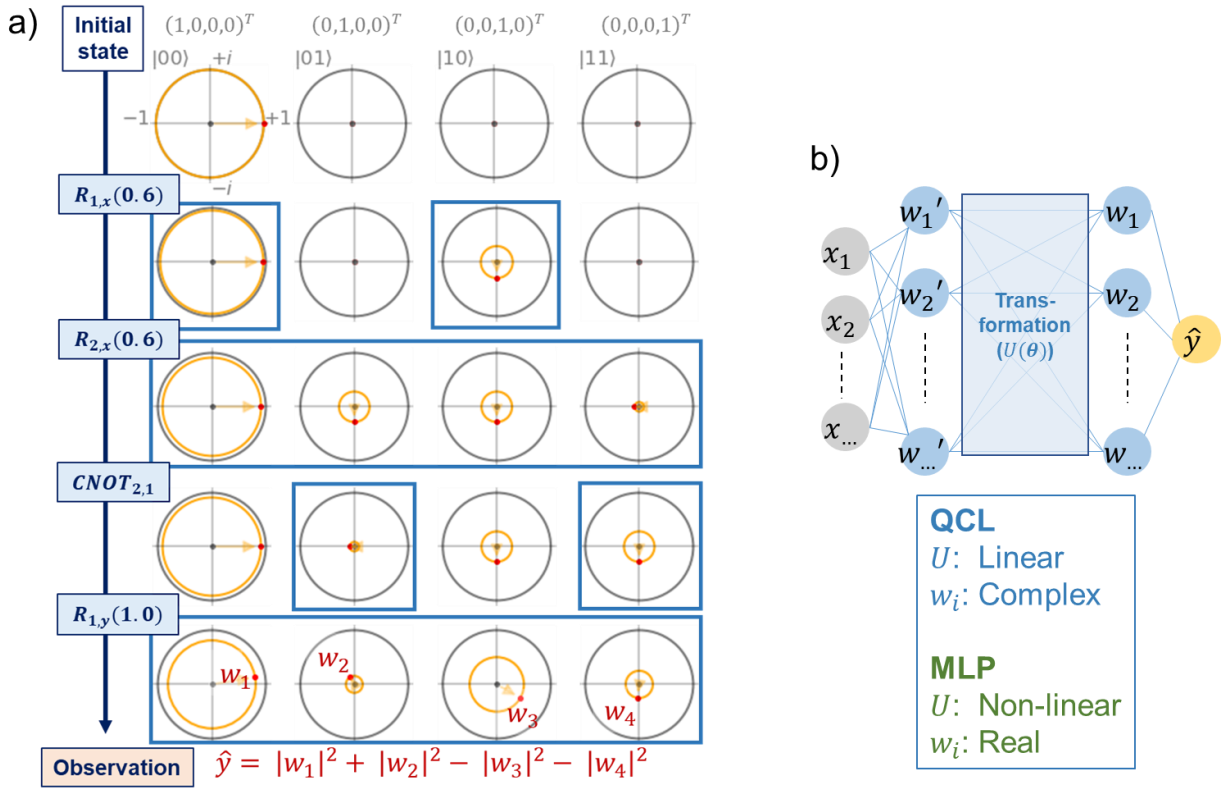


Figure S 5 a) Visualized state vector for an example circuit of $U(\theta)V(x) = R_{1,y}(\theta) \cdot \mathrm{CNOT}_{1,2} \cdot R_{2,x}(x) \cdot R_{1,x}(x)$ ($\theta = 1.0, x = 0.6$). Coordinate w_i against four bases $|00\rangle = (1,0,0,0)^T, \ldots, |11\rangle = (0,0,0,1)^T$ are plotted as red points on complex planes. Changes of w_i by gates are marked by blue squares. b) Model design of QCL and MLP.

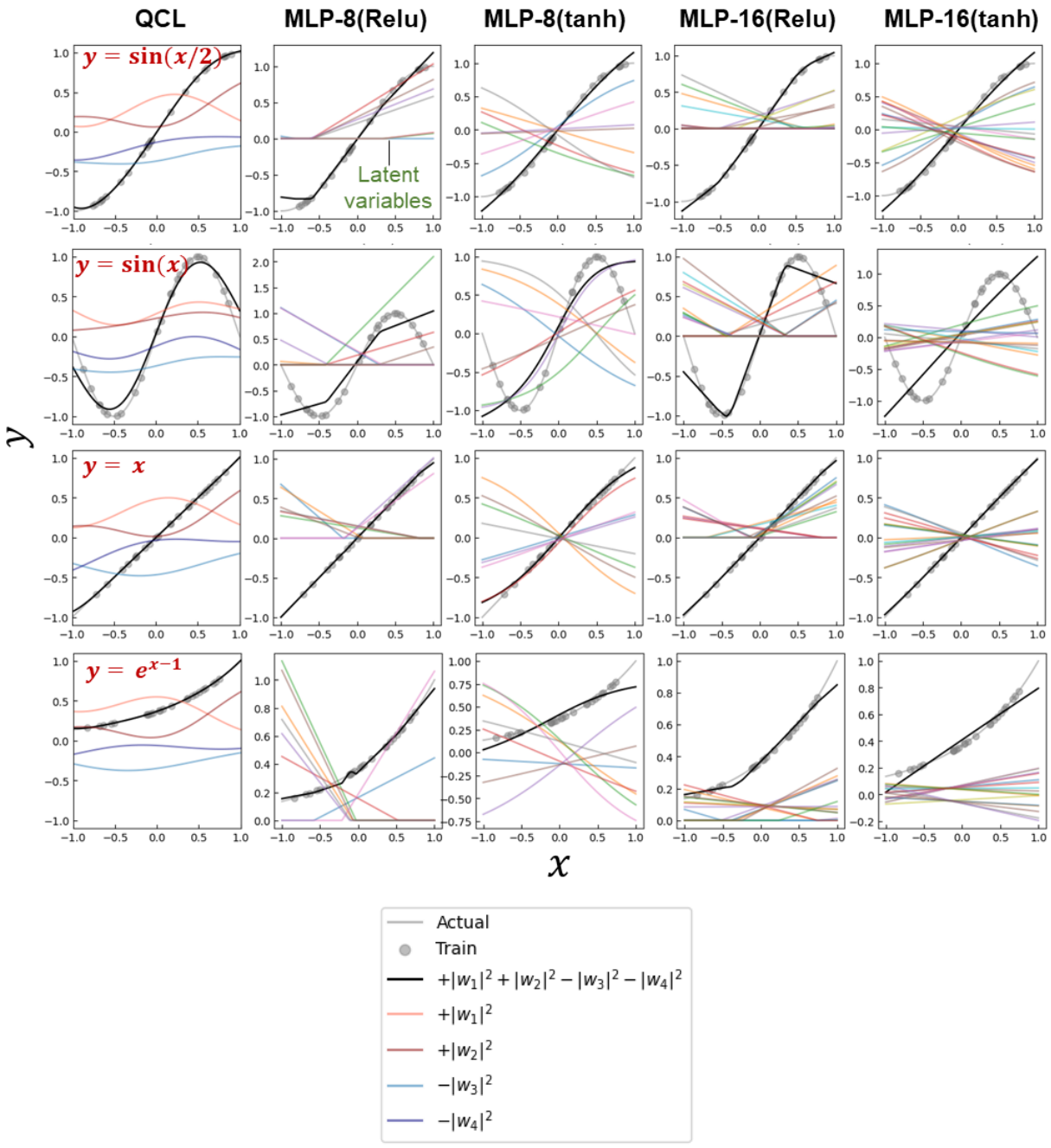
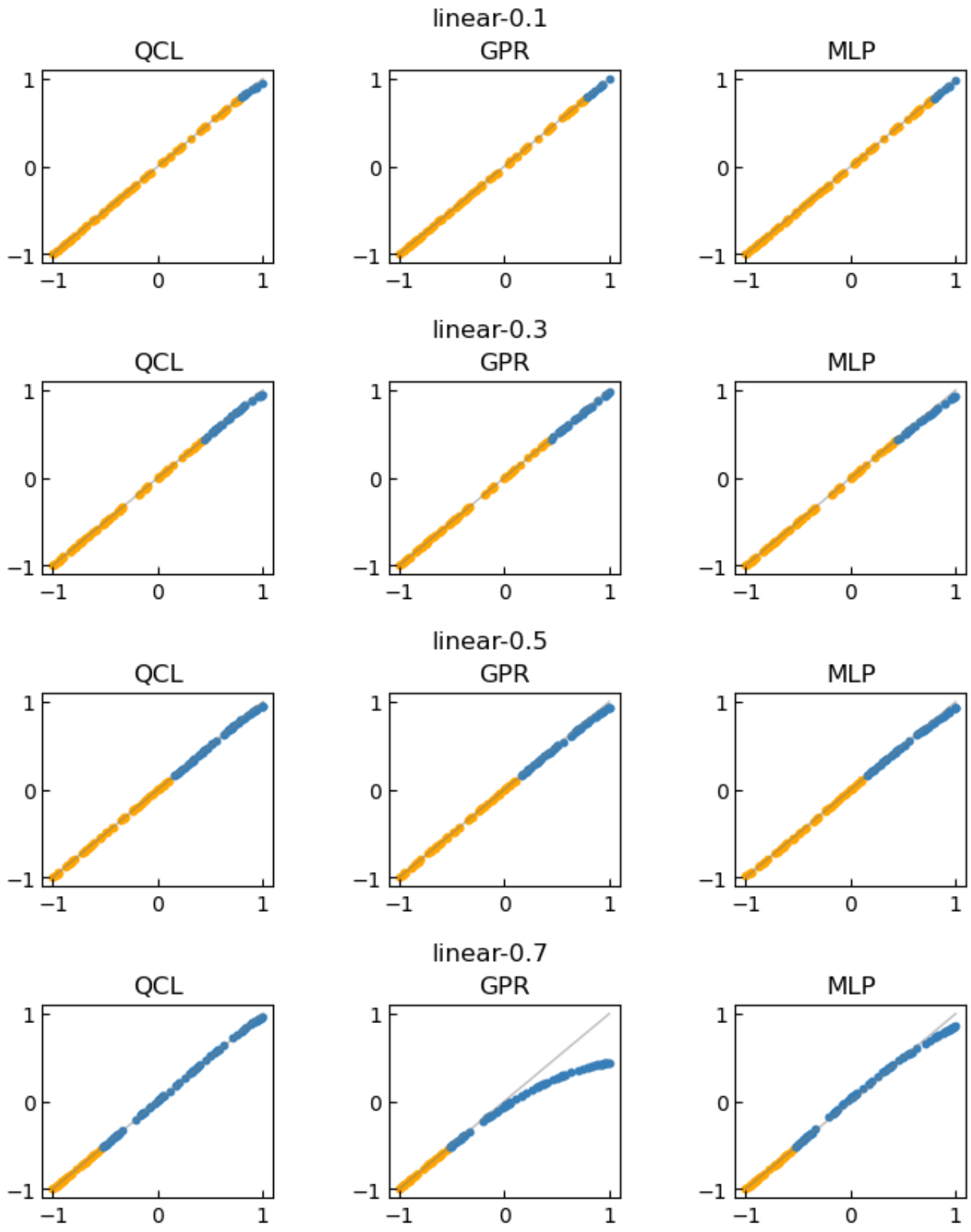
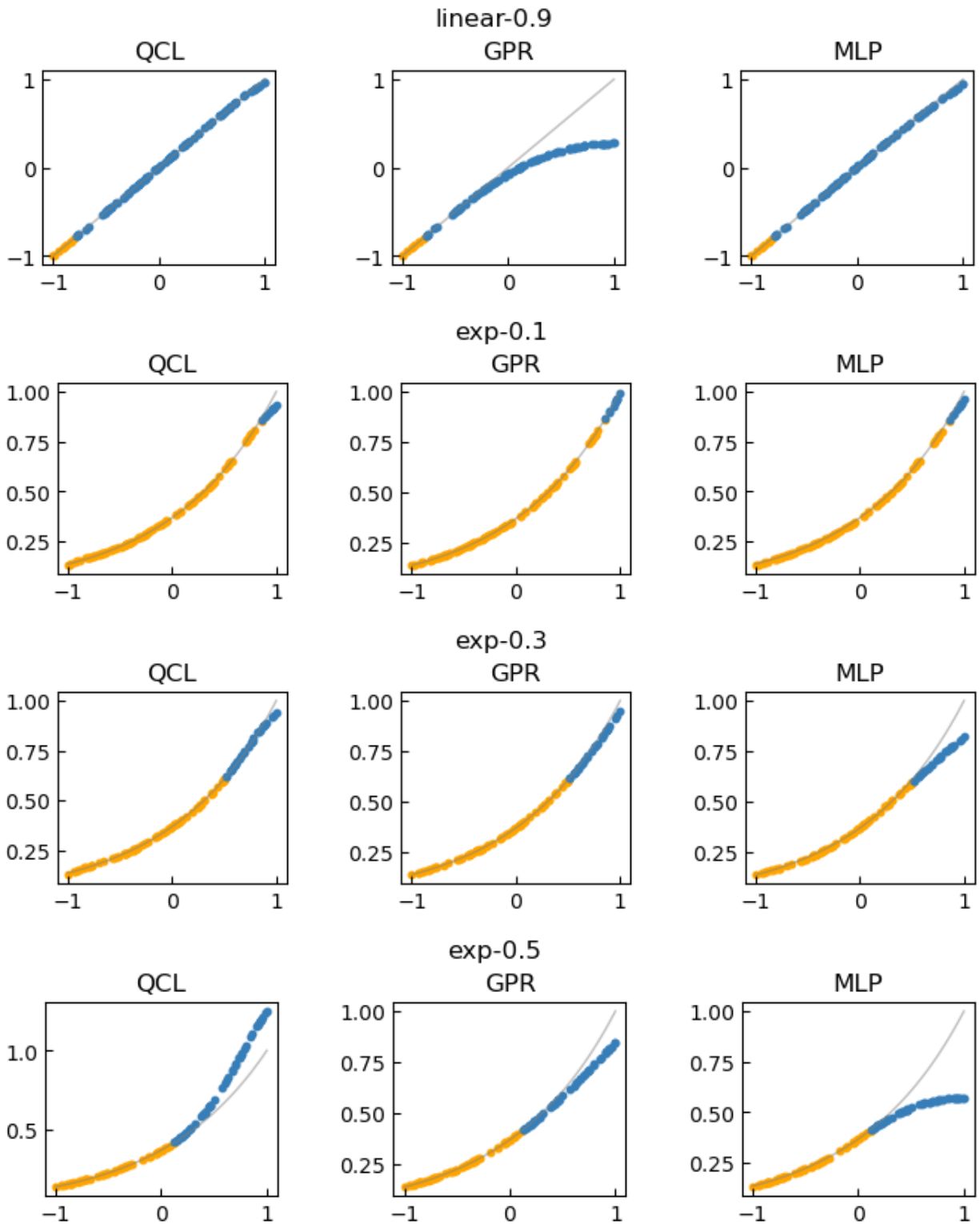
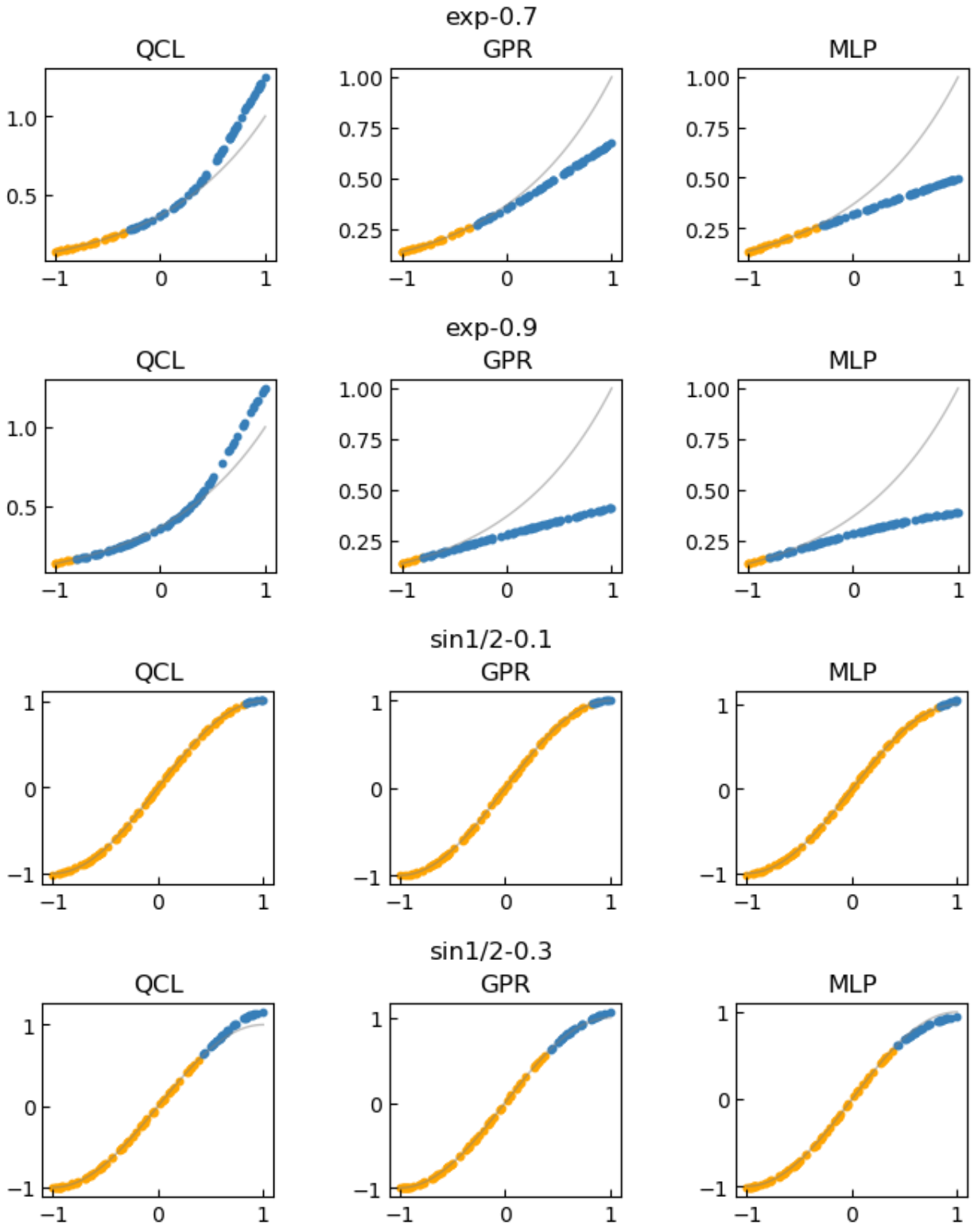
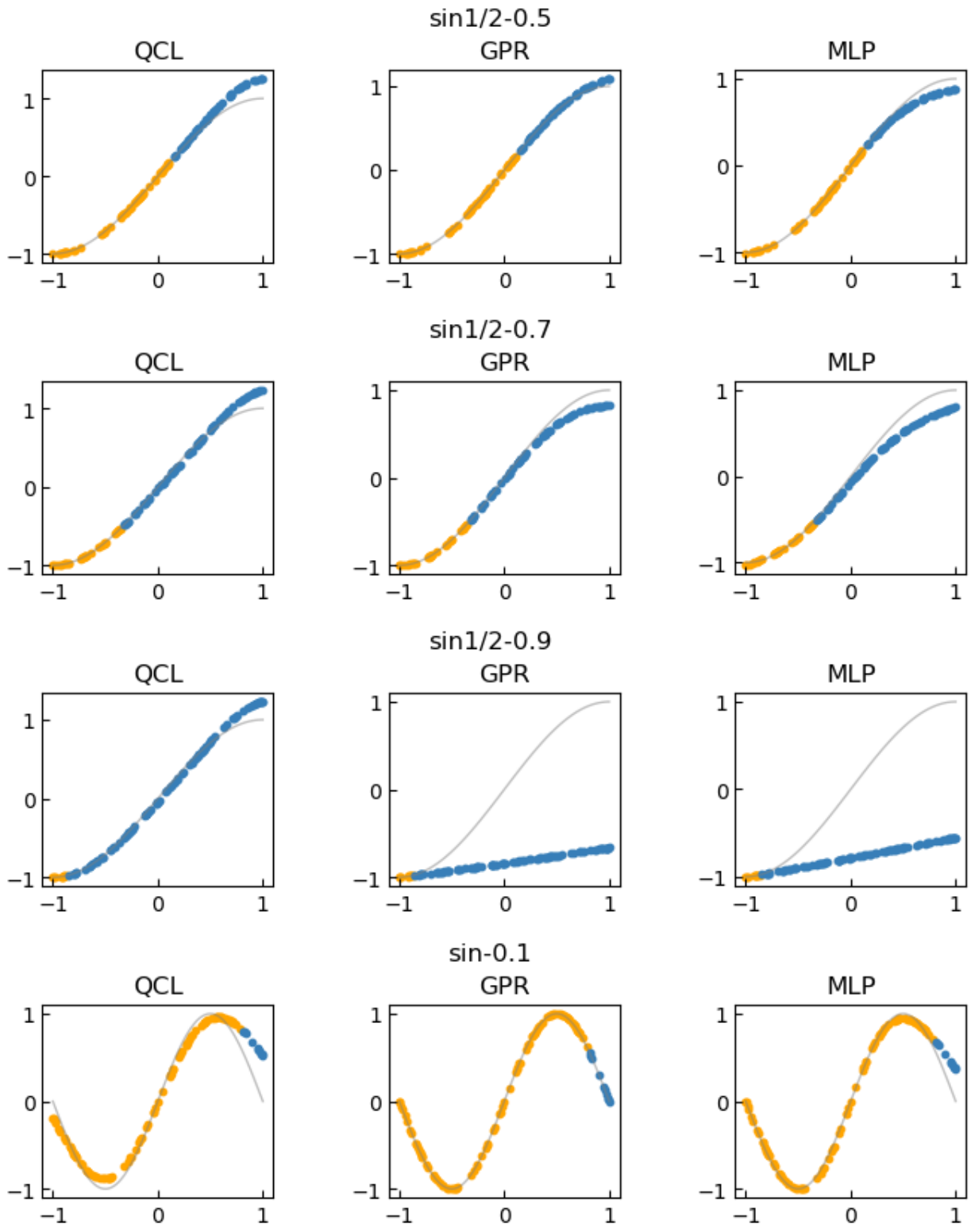


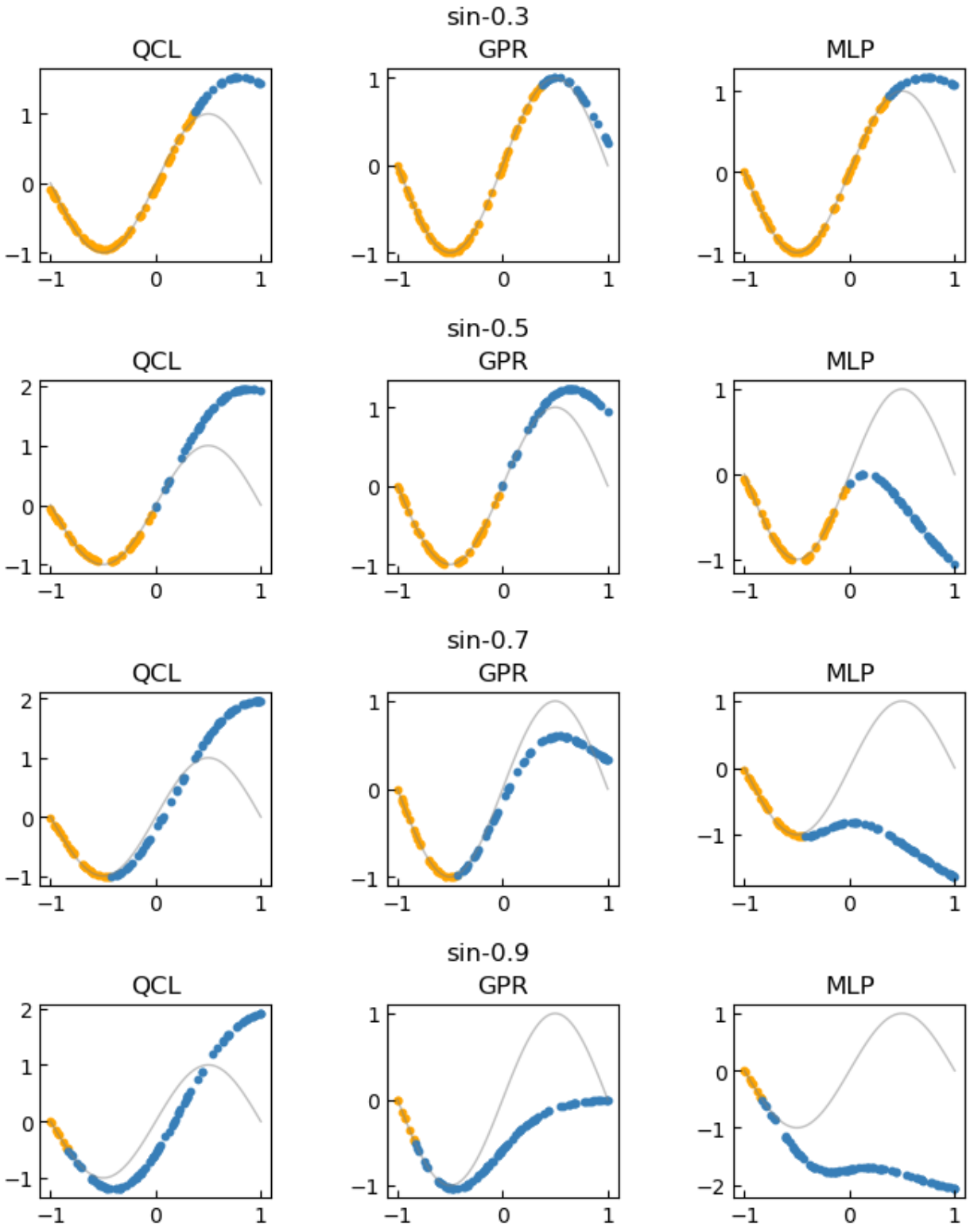
Figure S 6 Visualization of latent variables for the QCL and MLP models. Gray plots and lines show answer data, black lines correspond to final predictions, and other colored curves are latent variables. The expression of "MLP-8(ReLu)" represents that an 8-dimensional hidden layer and a ReLu activation function were selected as hyperparameters. Related data is shown in Figure 5.

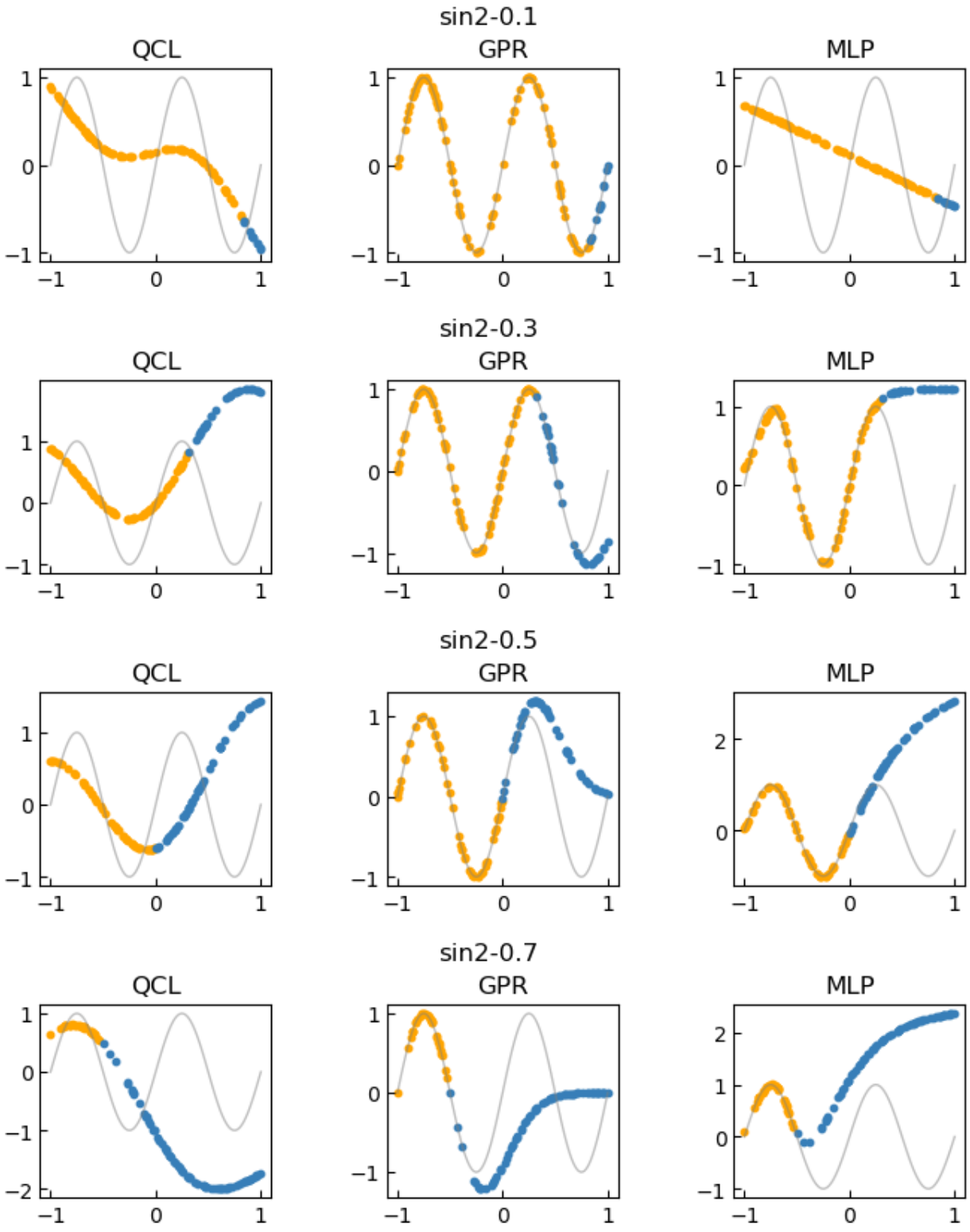

linear-0.1
QCL
GPR
MLP
linear-0.3
QCL
GPR
MLP
linear-0.5
QCL
GPR
MLP
linear-0.7
QCL
GPR
MLP
1
0
−1
−1
0
1


linear-0.9
QCL
GPR
MLP
exp-0.1
QCL
GPR
MLP
exp-0.3
QCL
GPR
MLP
exp-0.5
QCL
GPR
MLP


exp-0.7
QCL
GPR
MLP
exp-0.9
QCL
GPR
MLP
sin1/2-0.1
QCL
GPR
MLP
sin1/2-0.3
QCL
GPR
MLP


sin1/2-0.5
QCL
GPR
MLP
sin1/2-0.7
QCL
GPR
MLP
sin1/2-0.9
QCL
GPR
MLP
sin-0.1
QCL
GPR
MLP


sin-0.3
QCL
GPR
MLP
sin-0.5
QCL
GPR
MLP
sin-0.7
QCL
GPR
MLP
sin-0.9
QCL
GPR
MLP


sin2-0.1
QCL
GPR
MLP
sin2-0.3
QCL
GPR
MLP
sin2-0.5
QCL
GPR
MLP
sin2-0.7
QCL
GPR
MLP

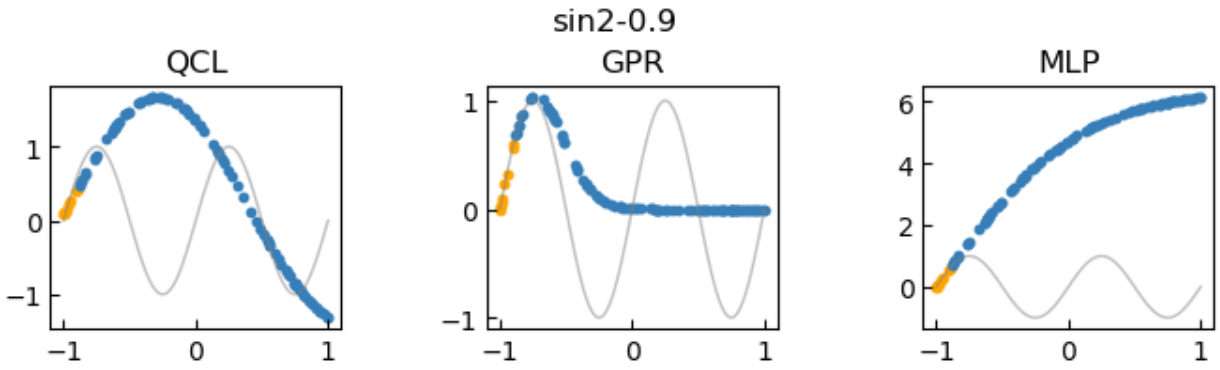


Figure S 7 Extrapolating predictions of linear, exponential, and sinusoidal functions by QCL, GPR (RBF), and MLP-8. Random 100 points were generated according to the original functions. Extrapolating 10, 30, 50, 70, or 90% of the data were selected as testing sets, and the rest were training. The expression of, e.g., "sin2-0.9" indicates that a $\sin 2x$ function was fitted with the 90% extrapolating testing data.

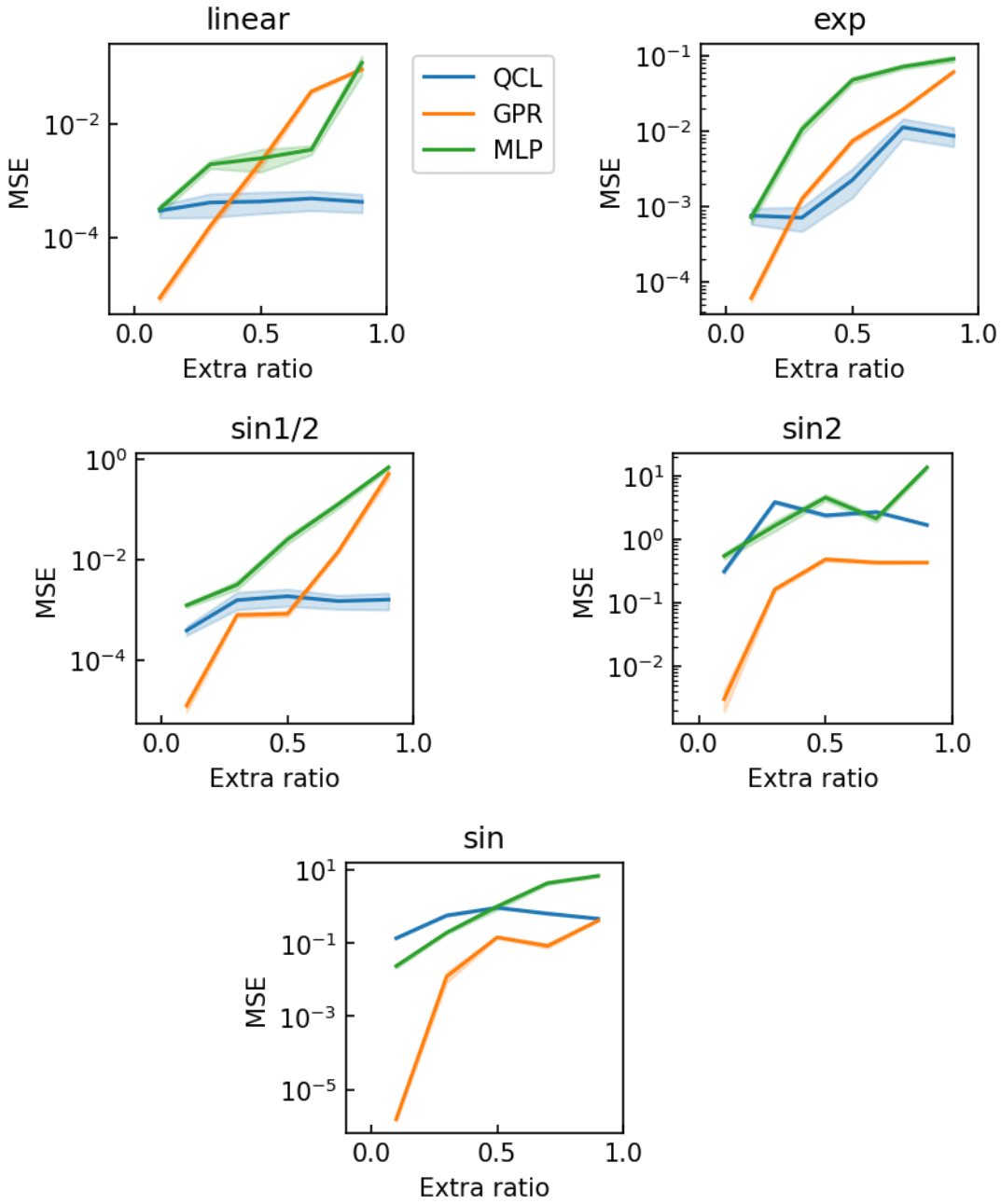


Figure S 8 Statistical extrapolating performances for Figure S 7. The random dataset preparation and fitting were repeated 30 times. Transparent regions show standard errors with 68% confidence intervals.

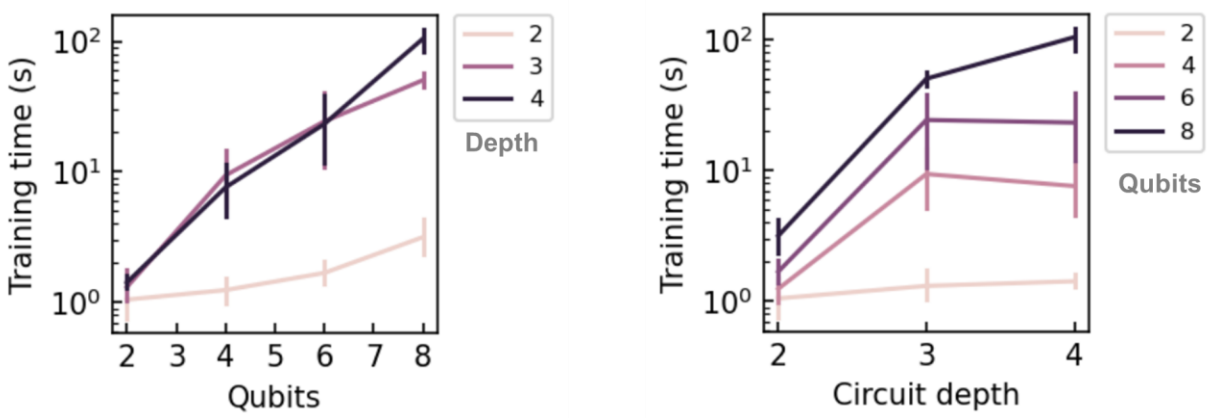


Figure S 9 Training time of QCL model (configuration shown in Figure 3b). The number of qubits n and circuit depth m were changed to train random 50 records of $y = \sin(x)$. Predictions were done by calculating from state vectors and repeated five times for each condition. Error bars indicate 95% confidence intervals assuming Gaussian distribution.

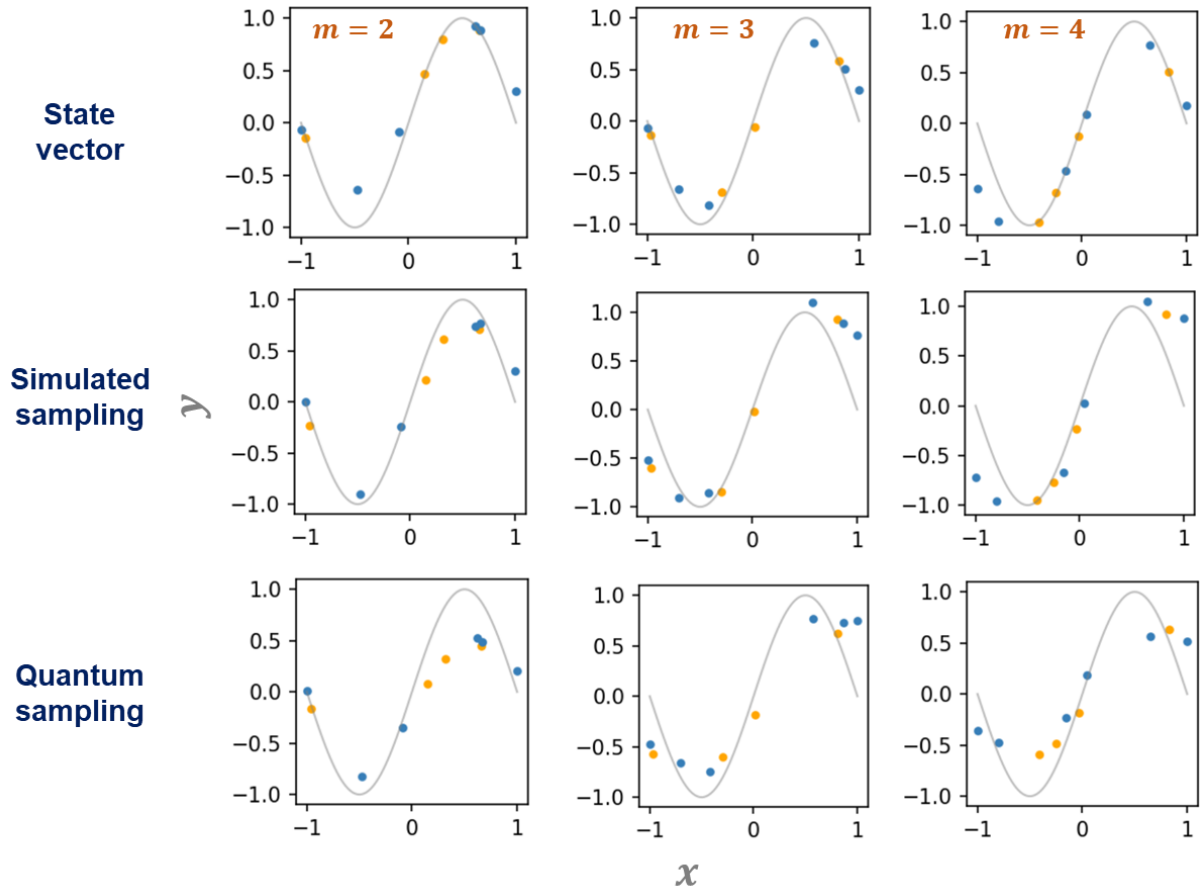


Figure S 10 Predicting the function of $y = \sin(x)$ by an actual quantum computer (IBM Quantum) with $m = 2, 3,$ or 4. Models were trained using the output of state vectors. Then, simulated or actual quantum computations were conducted to predict the same data from sampling results (**Eq 7**). For one record, sampling was done 1000 times. The accuracy of simulated sampling was worse than the state vector due to the randomness. Worse results of quantum sampling than simulation meant that noises during quantum computing affected the predictions.

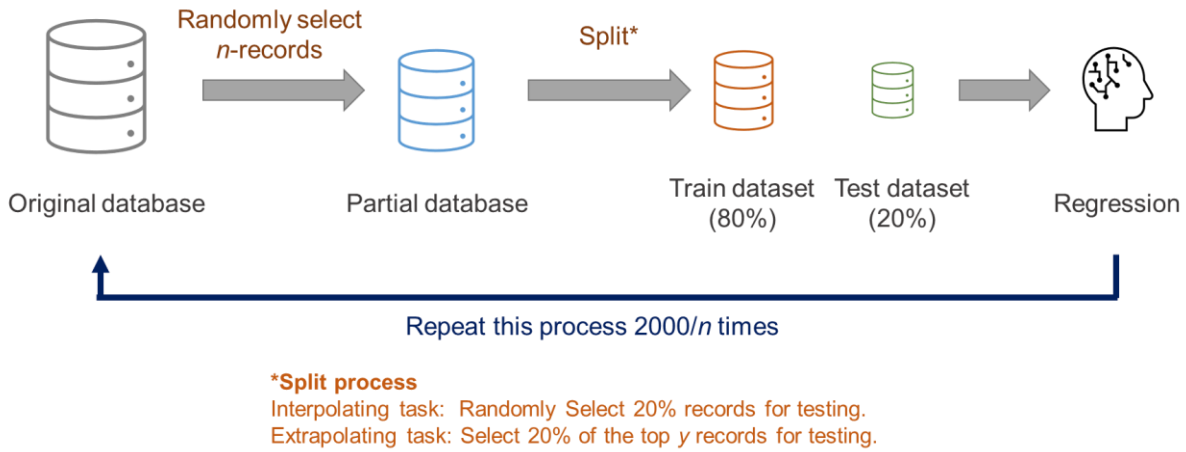
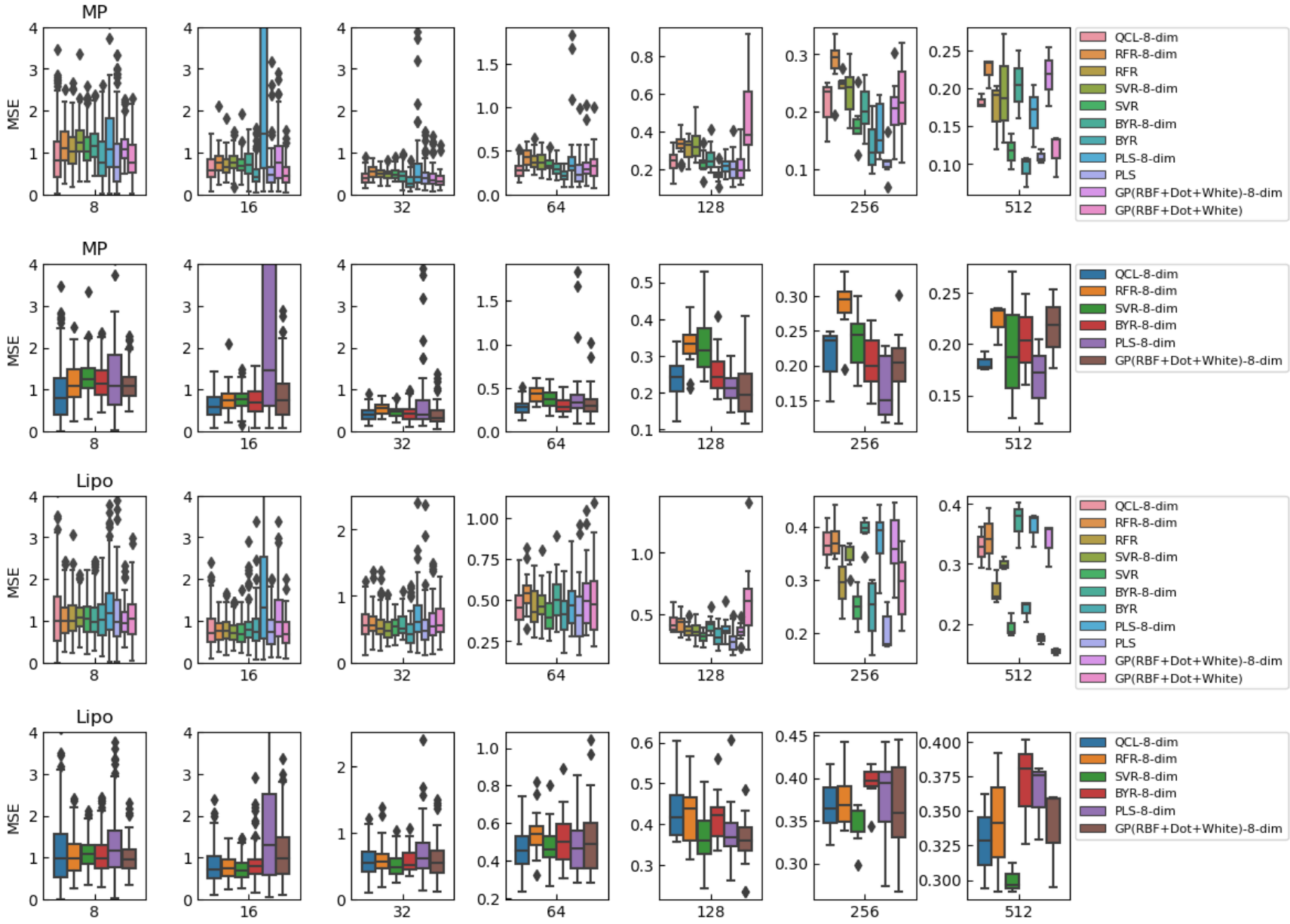
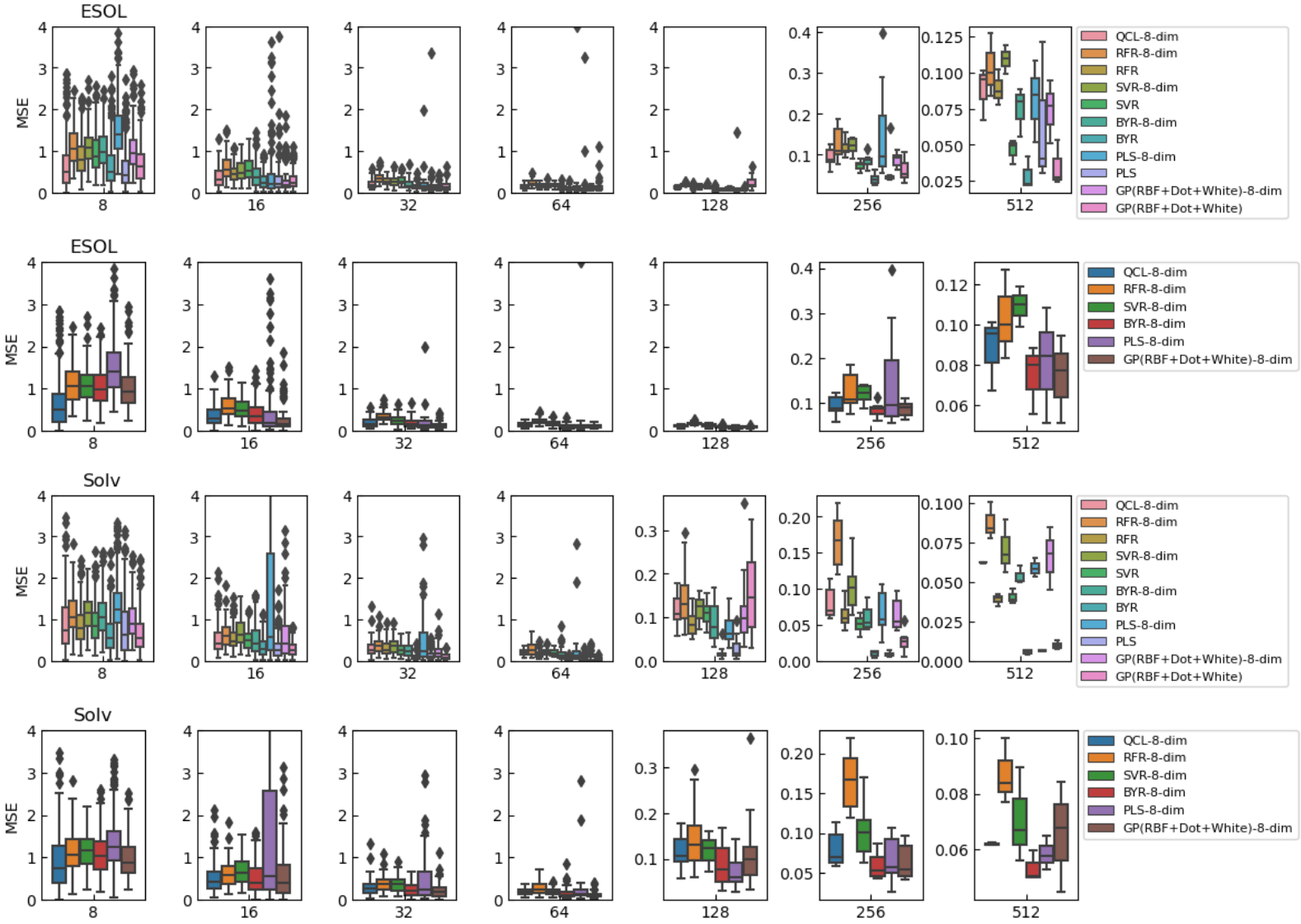
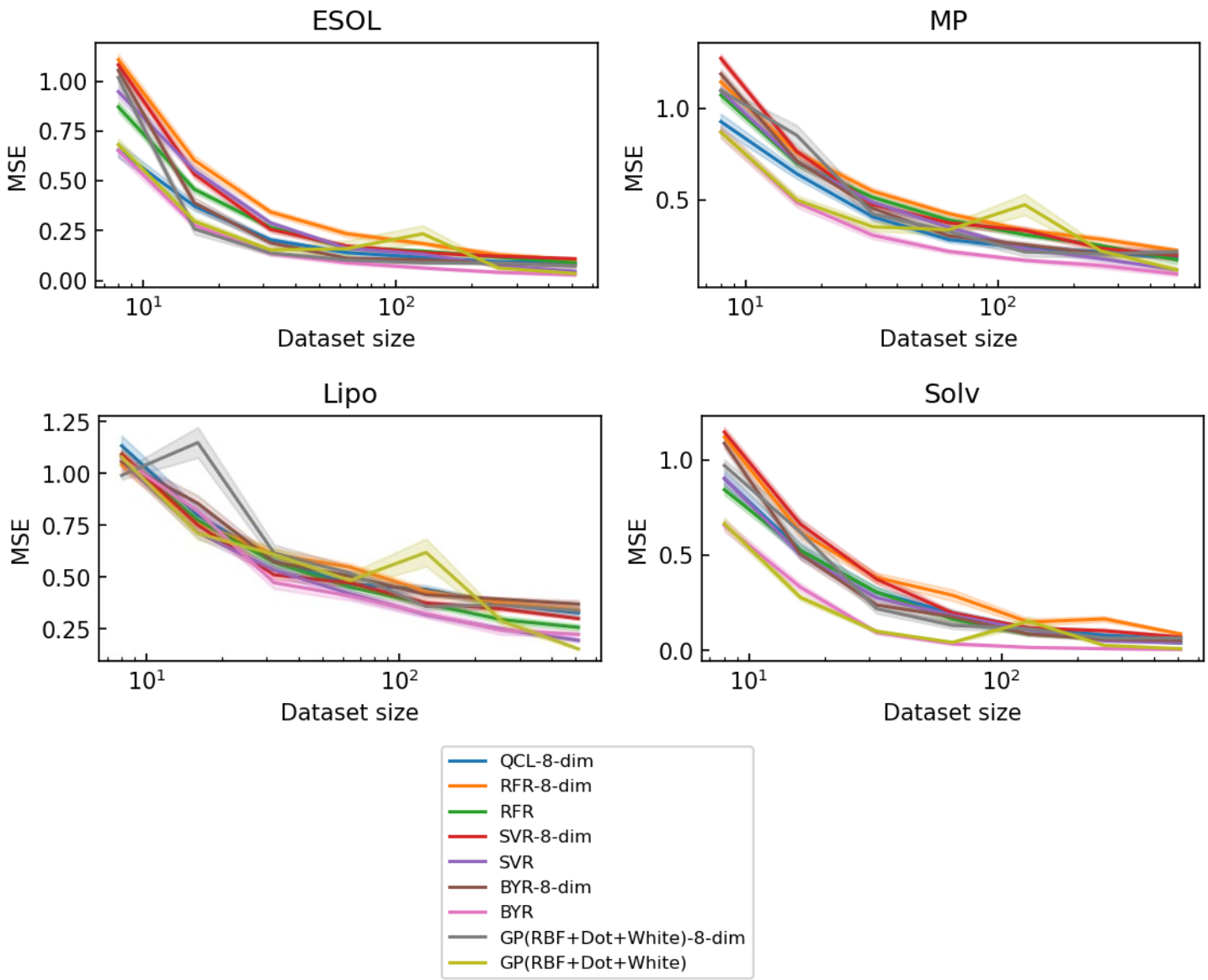


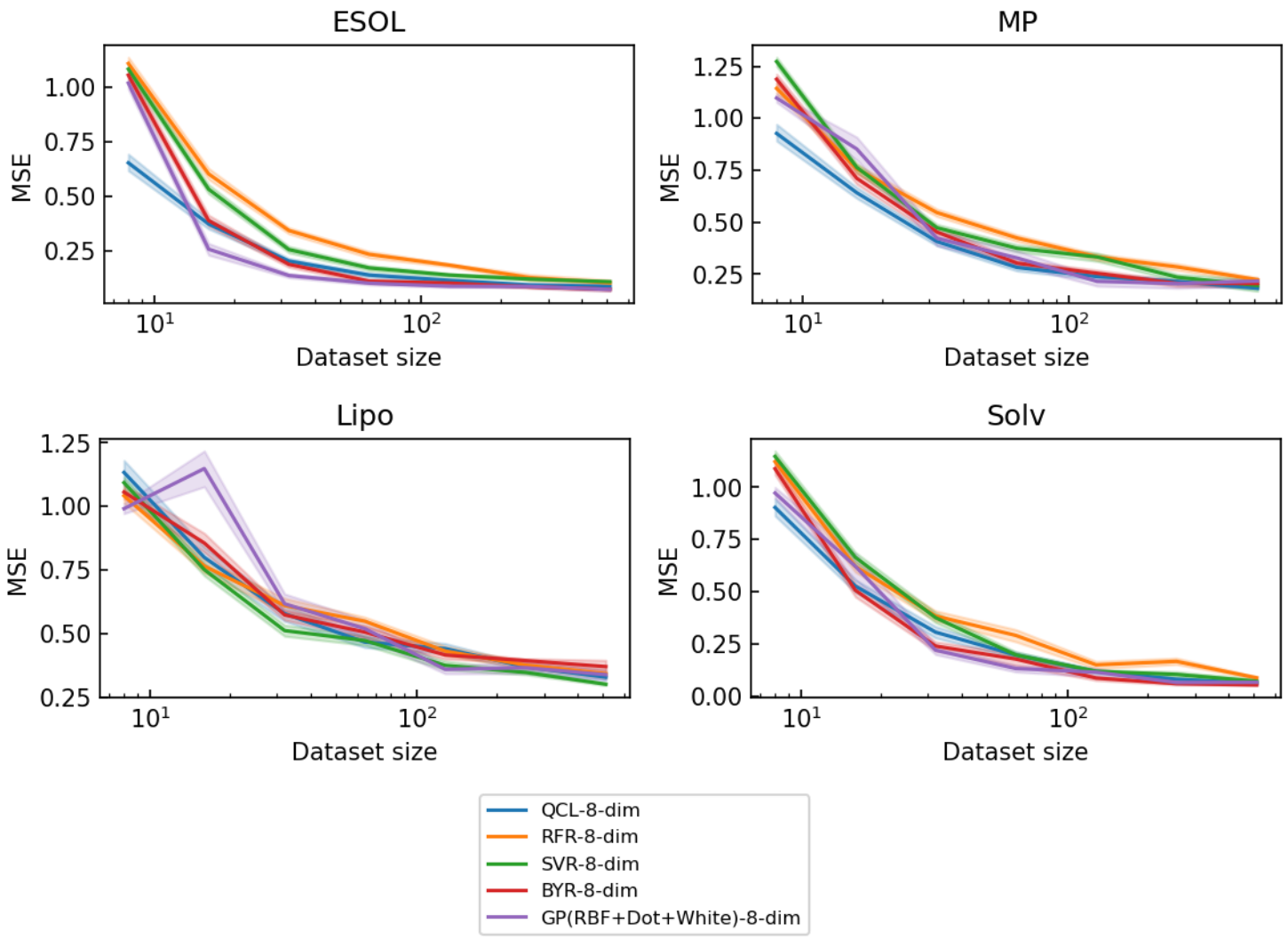
Figure S 11 Dataset preparation and regression steps for the molecular property prediction task. The dataset size of n was set to be 8, 16, 32, 64, 128, 256, or 512.


MP
MP
Lipo
Lipo
MSE
8
16
32
64
128
256
512
QCL-8-dim
RFR-8-dim
RFR
SVR-8-dim
SVR
BYR-8-dim
BYR
PLS-8-dim
PLS
GP(RBF+Dot+White)-8-dim
GP(RBF+Dot+White)


ESOL
ESOL
Solv
Solv
MSE
Dataset size
QCL-8-dim
RFR-8-dim
RFR
SVR-8-dim
SVR
BYR-8-dim
BYR
PLS-8-dim
PLS
GP(RBF+Dot+White)-8-dim
GP(RBF+Dot+White)

a)

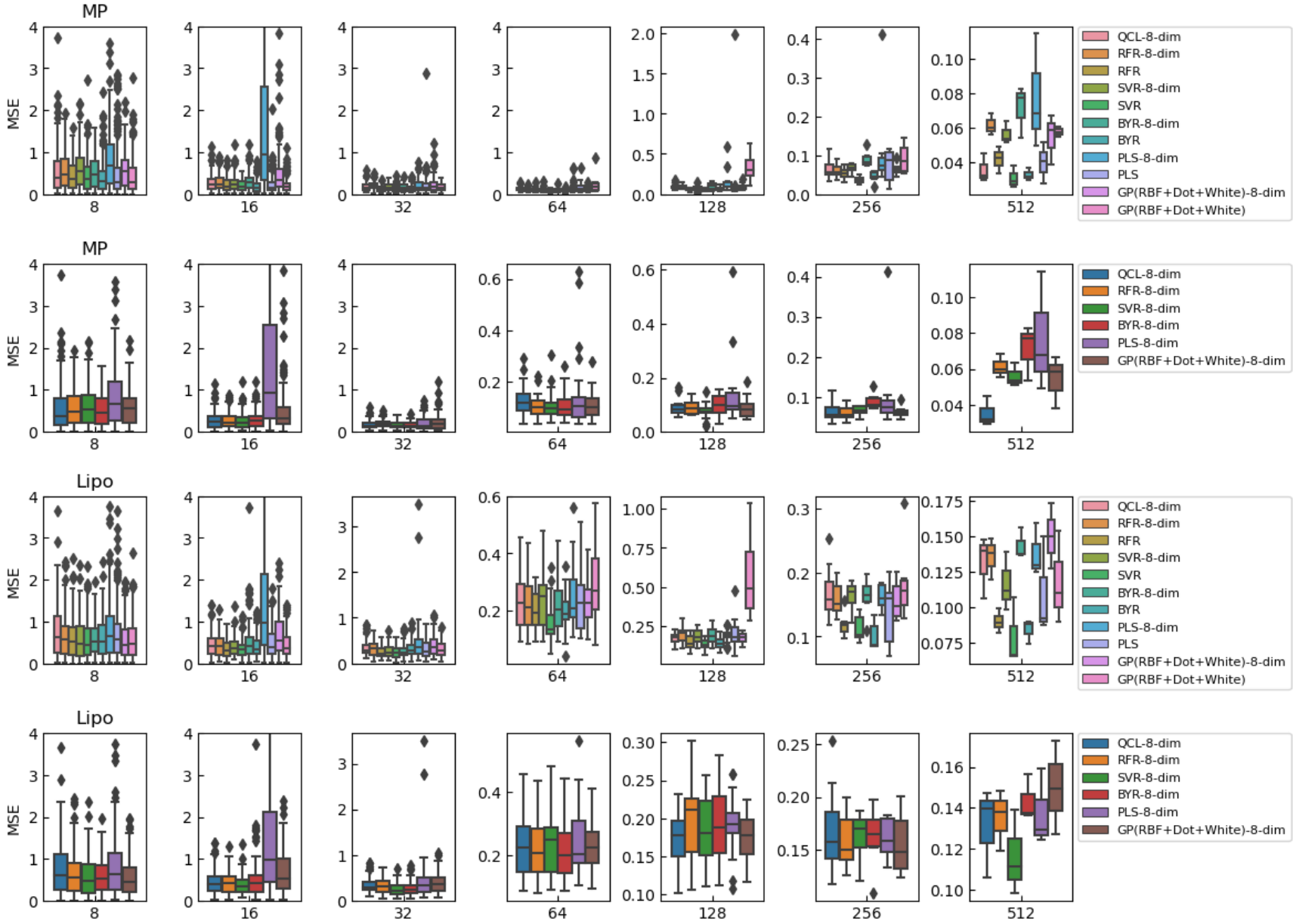

ESOL
MP
Lipo
Solv
MSE
Dataset size
10^1
10^2
QCL-8-dim
RFR-8-dim
RFR
SVR-8-dim
SVR
BYR-8-dim
BYR
GP(RBF+Dot+White)-8-dim
GP(RBF+Dot+White)

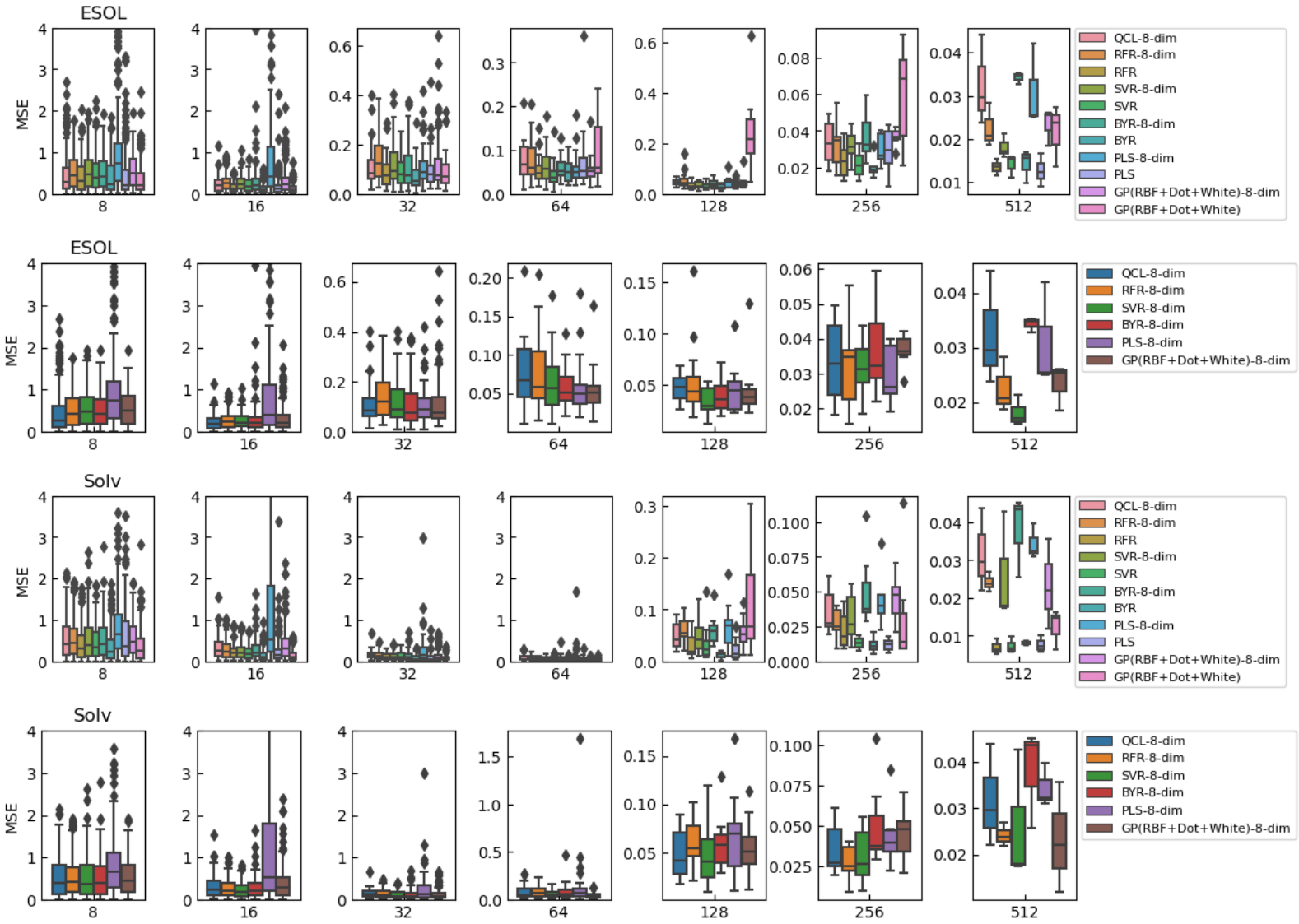


b)

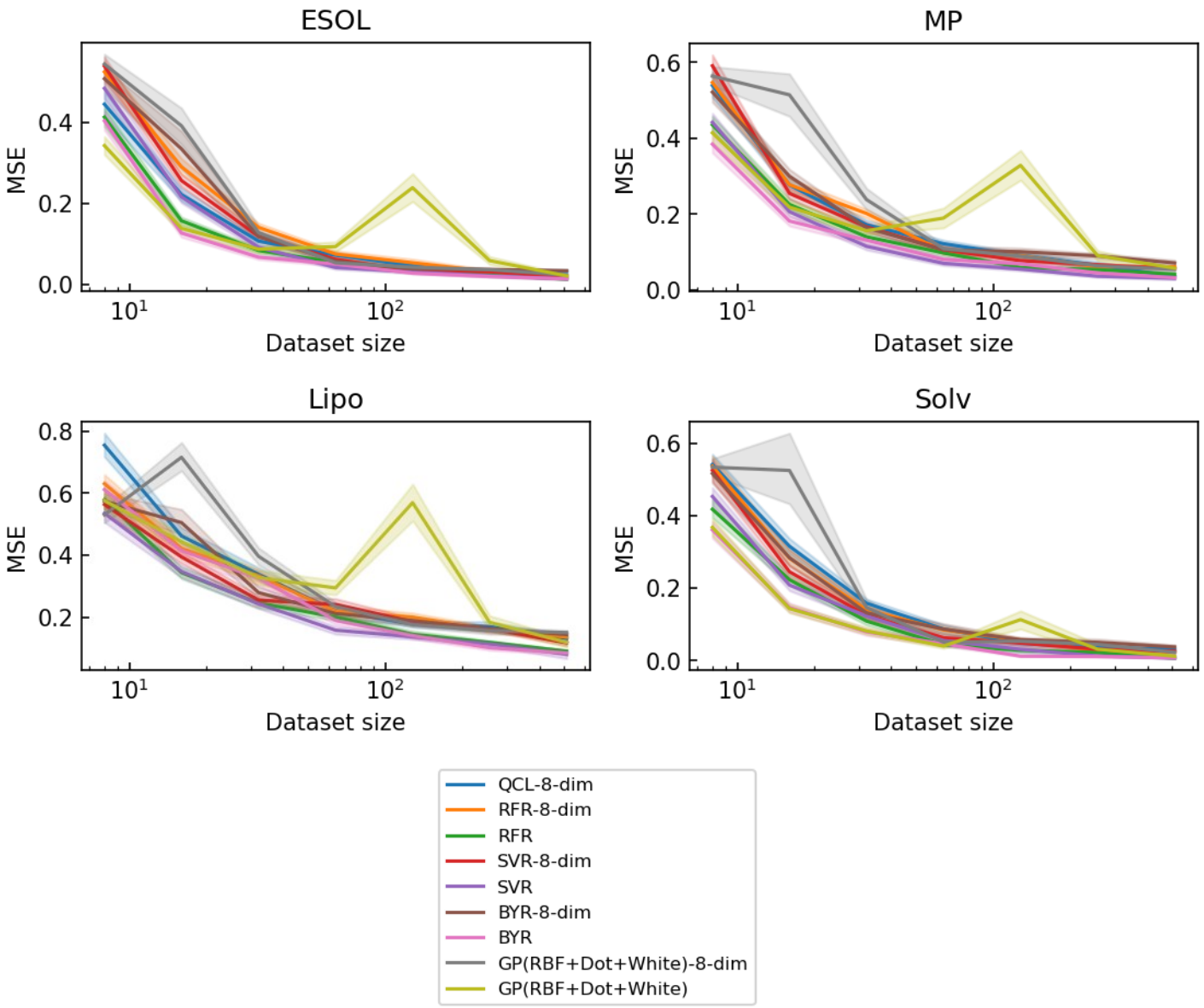
Figure S 12 Regression results for the extrapolating tasks, with lipophilicity (Lipo), hydration free energy of small molecules in water (Solv), log solubility in water (ESOL), and melting point (MP) datasets. a) Box plots. b) Line plots with standard errors with 68%

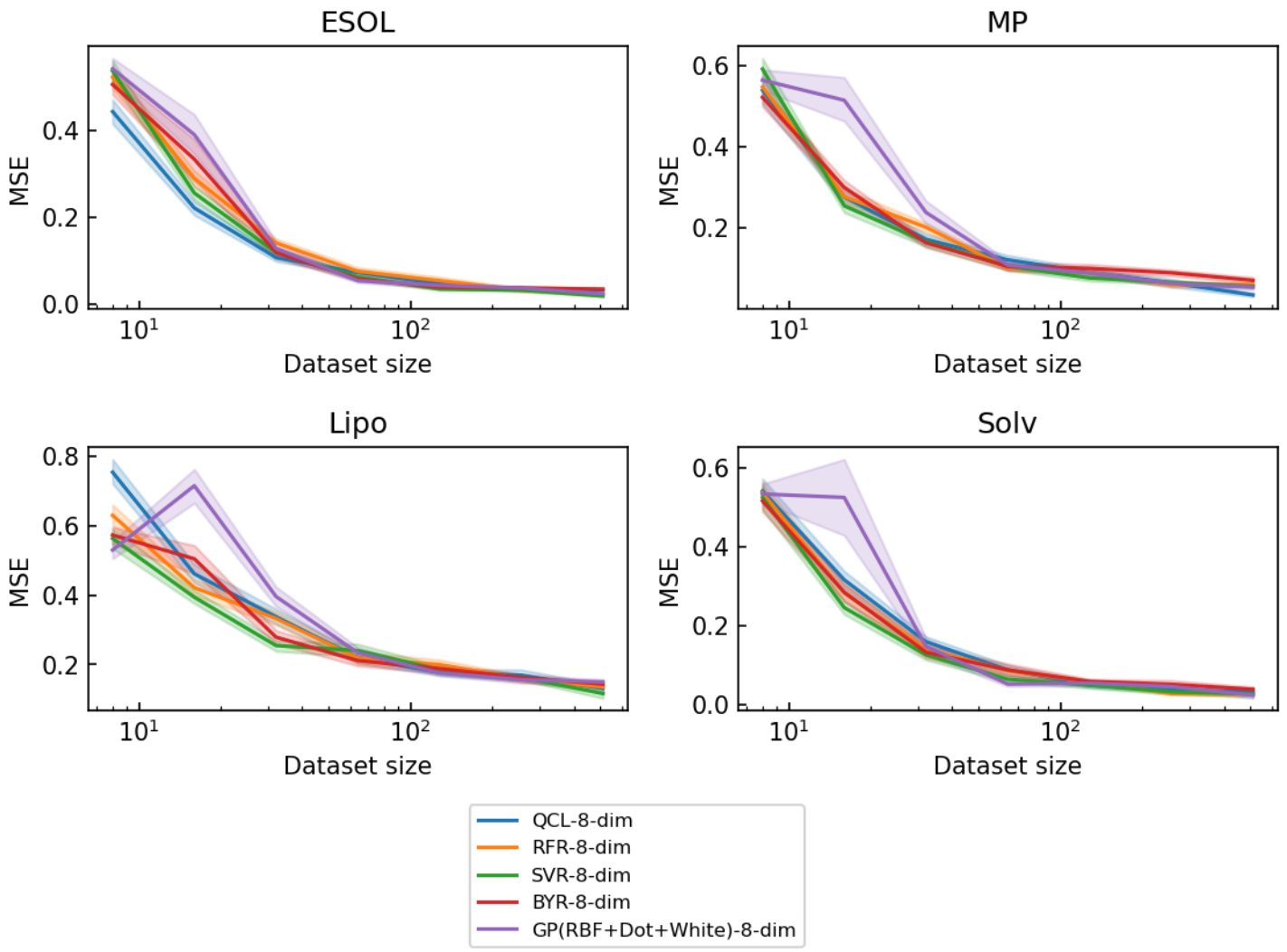
confidence intervals. In the legends, “8-dim” means that the explanatory variables were compressed from about 200- to 8-dimensional by principal component analysis.


MP
MP
Lipo
Lipo
MSE
8
16
32
64
128
256
512
QCL-8-dim
RFR-8-dim
RFR
SVR-8-dim
SVR
BYR-8-dim
BYR
PLS-8-dim
PLS
GP(RBF+Dot+White)-8-dim
GP(RBF+Dot+White)


ESOL
ESOL
Solv
Solv
MSE
Dataset size
QCL-8-dim
RFR-8-dim
RFR
SVR-8-dim
SVR
BYR-8-dim
BYR
PLS-8-dim
PLS
GP(RBF+Dot+White)-8-dim
GP(RBF+Dot+White)

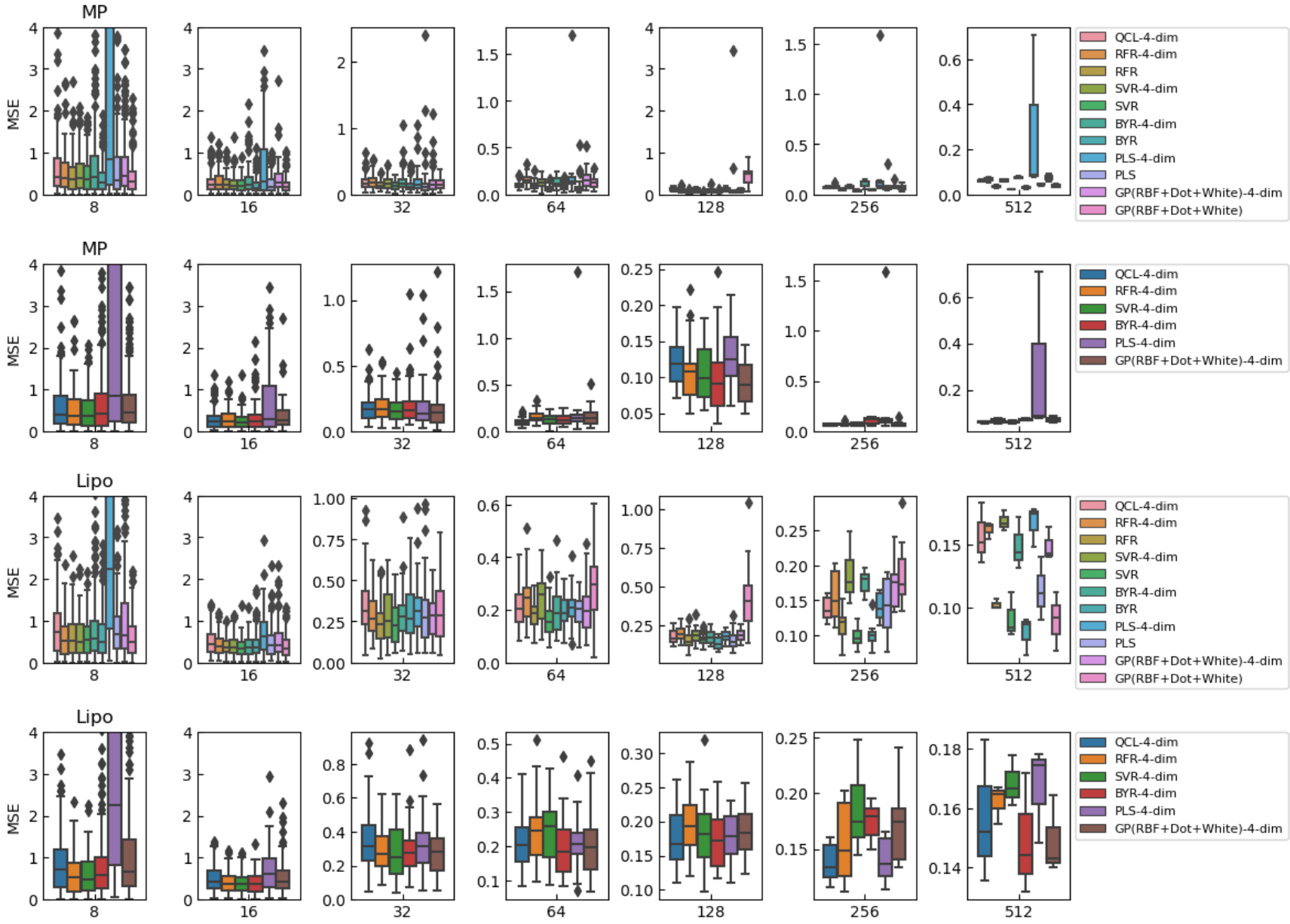
a)

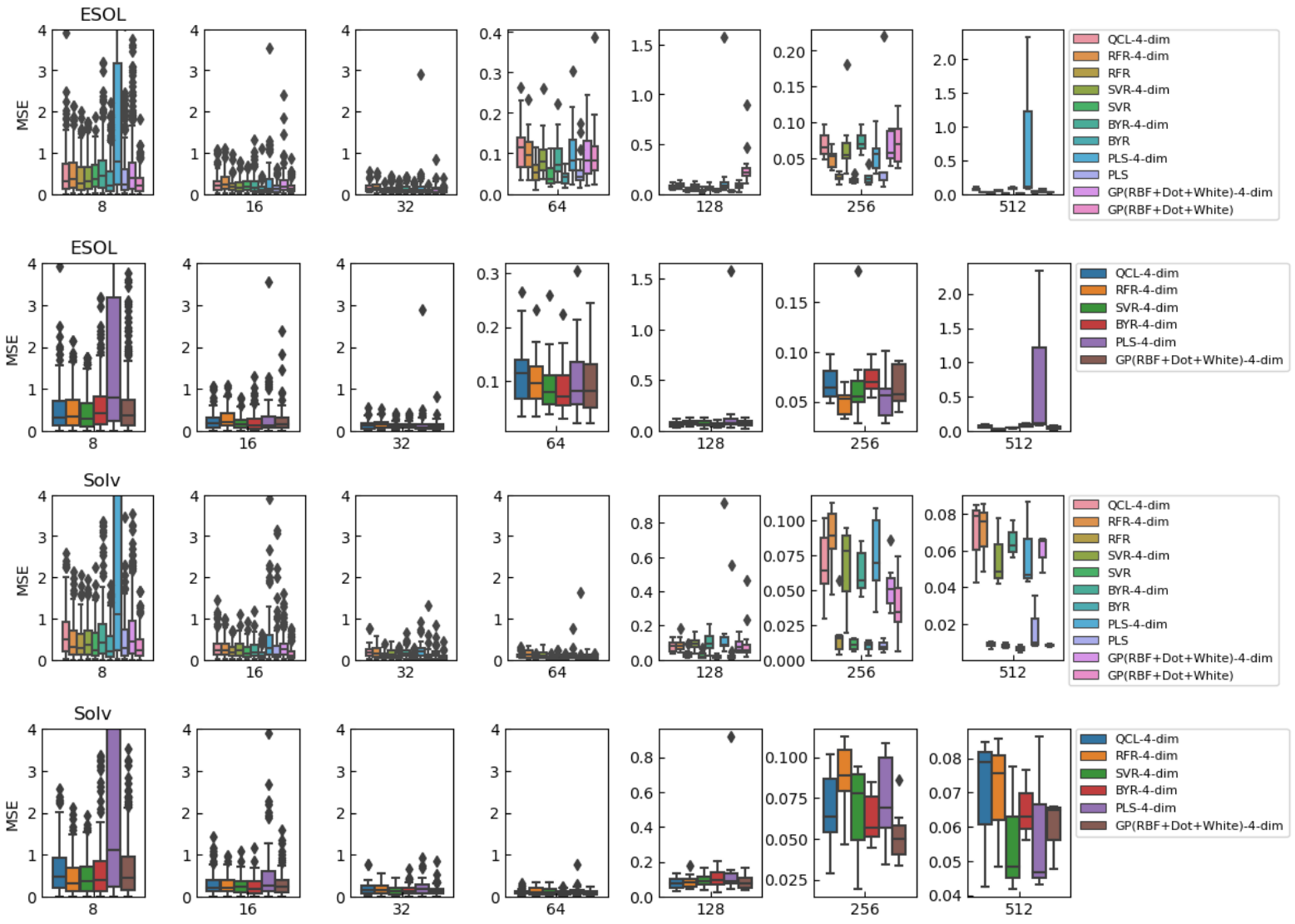

ESOL
MP
Lipo
Solv
MSE
Dataset size
10^1
10^2
QCL-8-dim
RFR-8-dim
RFR
SVR-8-dim
SVR
BYR-8-dim
BYR
GP(RBF+Dot+White)-8-dim
GP(RBF+Dot+White)



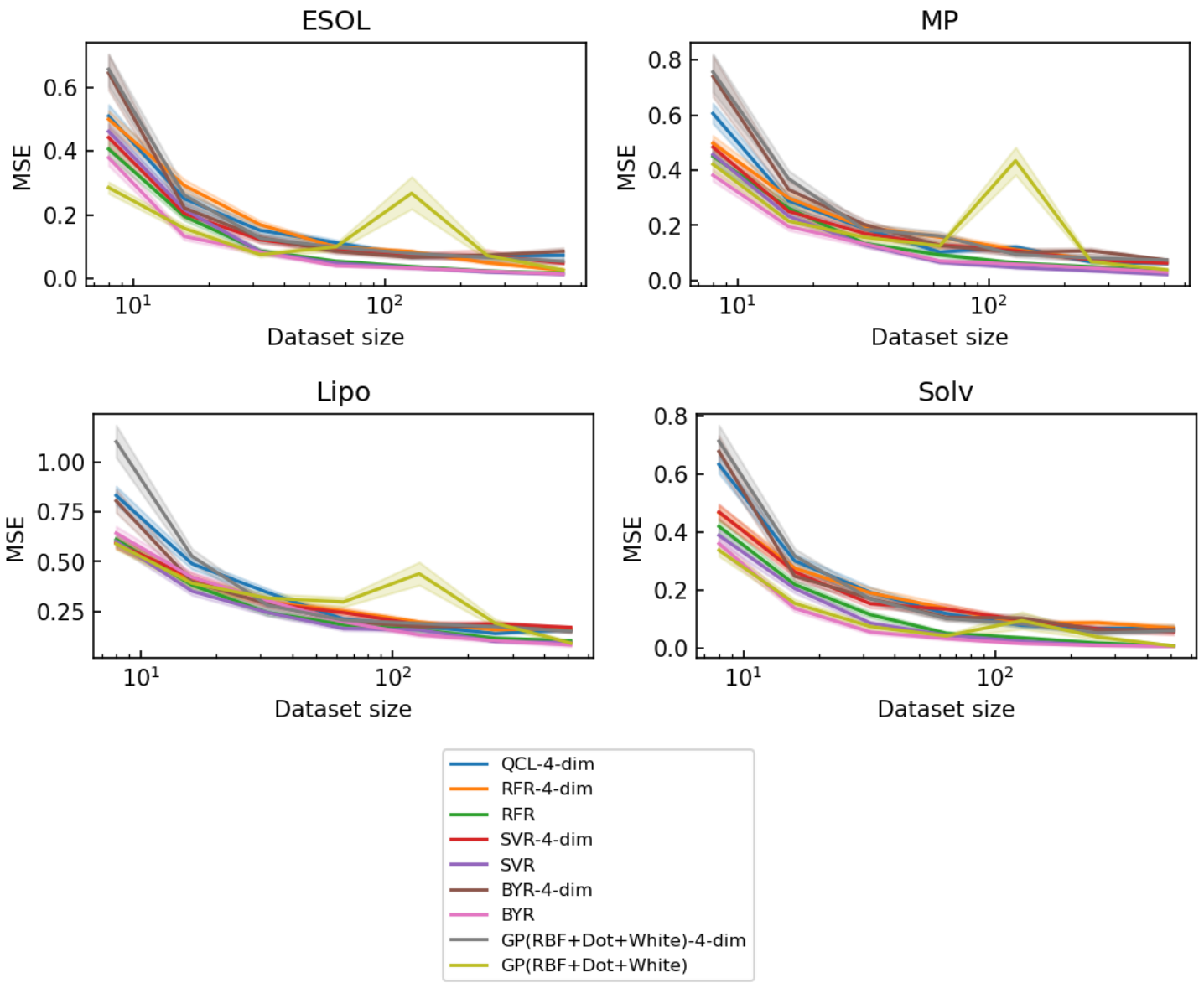
b)

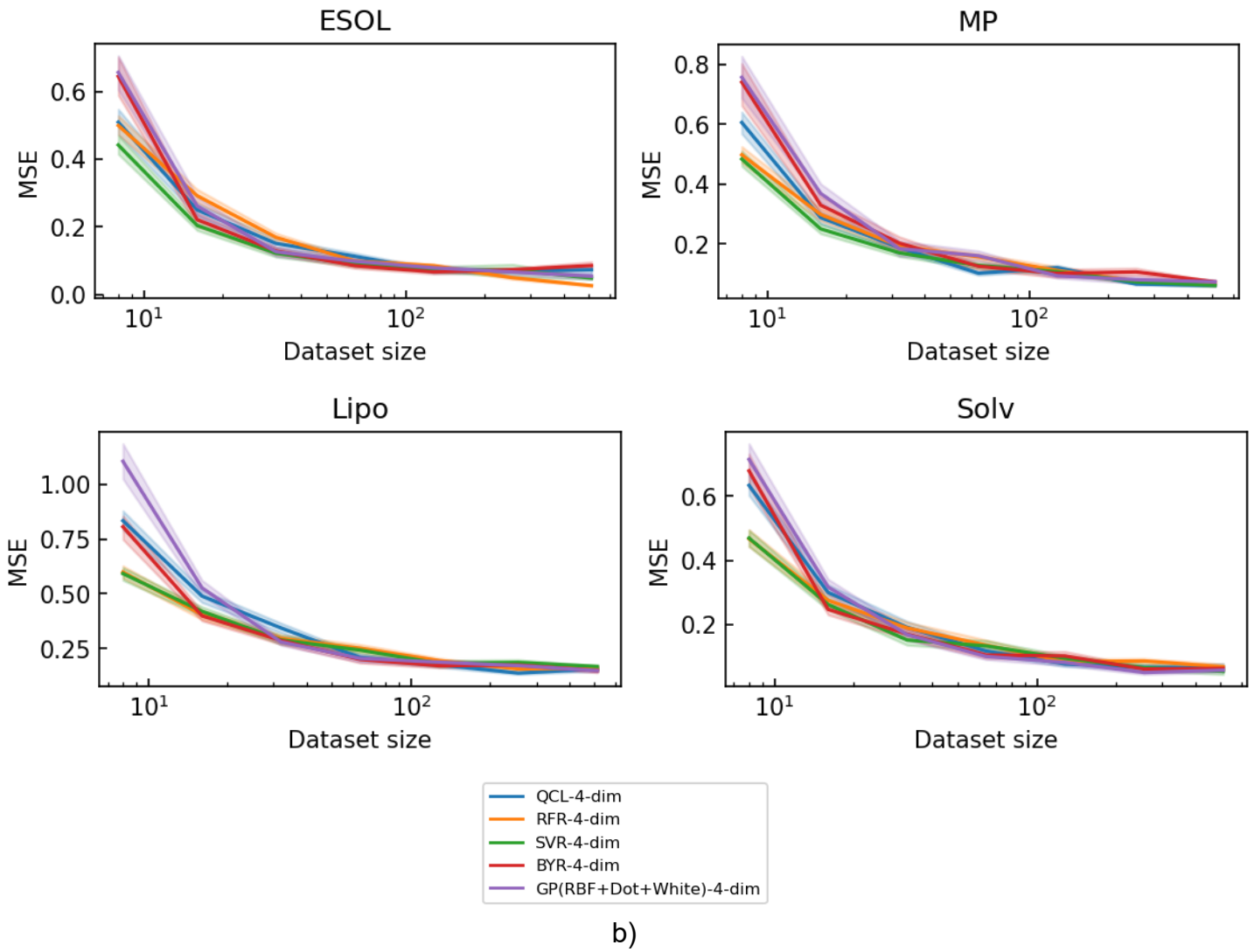
Figure S 13 Regression results for the interpolating tasks as Box plots. b) Line plots with standard errors with 68% confidence intervals. In the figures, 20% of testing data were randomly sampled from the dataset, whereas the top 20% records of y were extracted in Figure S 12.


MP
MP
Lipo
Lipo
MSE
8
16
32
64
128
256
512
QCL-4-dim
RFR-4-dim
RFR
SVR-4-dim
SVR
BYR-4-dim
BYR
PLS-4-dim
PLS
GP(RBF+Dot+White)-4-dim
GP(RBF+Dot+White)


ESOL
MSE
8
16
32
64
128
256
512
QCL-4-dim
RFR-4-dim
RFR
SVR-4-dim
SVR
BYR-4-dim
BYR
PLS-4-dim
PLS
GP(RBF+Dot+White)-4-dim
GP(RBF+Dot+White)
ESOL
MSE
QCL-4-dim
RFR-4-dim
SVR-4-dim
BYR-4-dim
PLS-4-dim
GP(RBF+Dot+White)-4-dim
Solv
MSE
QCL-4-dim
RFR-4-dim
RFR
SVR-4-dim
SVR
BYR-4-dim
BYR
PLS-4-dim
PLS
GP(RBF+Dot+White)-4-dim
GP(RBF+Dot+White)
Solv
MSE
QCL-4-dim
RFR-4-dim
SVR-4-dim
BYR-4-dim
PLS-4-dim
GP(RBF+Dot+White)-4-dim
Dataset size

a)


ESOL
MP
Lipo
Solv
MSE
Dataset size
10^1
10^2
0.0
0.2
0.4
0.6
0.8
0.25
0.50
0.75
1.00
QCL-4-dim
RFR-4-dim
RFR
SVR-4-dim
SVR
BYR-4-dim
BYR
GP(RBF+Dot+White)-4-dim
GP(RBF+Dot+White)



b)

Figure S 14 Regression results for the interpolating tasks with 4-dimensional vectors. a) Box plots and b) Line plots with standard errors with 68% confidence intervals.. A QCL circuit ($n = 8$) inputted a vector of $(x_1, x_1, x_2, x_2, \ldots, x_4, x_4)$.

Table S4 List of molecular descriptors calculated by RDKit.

Name			
MaxEStateIndex	PEOE_VSA2	VSA_EState9	fr_aryl_methyl
MinEStateIndex	PEOE_VSA3	FractionCSP3	fr_azide
MaxAbsEStateIndex	PEOE_VSA4	HeavyAtomCount	fr_azo
MinAbsEStateIndex	PEOE_VSA5	NHOHCount	fr_barbitur
qed	PEOE_VSA6	NOCount	fr_benzene
MolWt	PEOE_VSA7	NumAliphaticCarbocycles	fr_benzodiazepine
HeavyAtomMolWt	PEOE_VSA8	NumAliphaticHeterocycles	fr_bicyclic
ExactMolWt	PEOE_VSA9	NumAliphaticRings	fr_diazo
NumValenceElectrons	SMR_VSA1	NumAromaticCarbocycles	fr_dihydropyridine
NumRadicalElectrons	SMR_VSA10	NumAromaticHeterocycles	fr_epoxide
MaxPartialCharge	SMR_VSA2	NumAromaticRings	fr_ester
MinPartialCharge	SMR_VSA3	NumHAcceptors	fr_ether
MaxAbsPartialCharge	SMR_VSA4	NumHDonors	fr_furan
MinAbsPartialCharge	SMR_VSA5	NumHeteroatoms	fr_guanido
FpDensityMorgan1	SMR_VSA6	NumRotatableBonds	fr_halogen
FpDensityMorgan2	SMR_VSA7	NumSaturatedCarbocycles	fr_hdrzine
FpDensityMorgan3	SMR_VSA8	NumSaturatedHeterocycles	fr_hdrzone
BCUT2D_MWHI	SMR_VSA9	NumSaturatedRings	fr_imidazole
BCUT2D_MWLOW	SlogP_VSA1	RingCount	fr_imide
BCUT2D_CHGHI	SlogP_VSA10	MolLogP	fr_isocyan
BCUT2D_CHGLO	SlogP_VSA11	MolMR	fr_isothiocyan
BCUT2D_LOGPHI	SlogP_VSA12	fr_Al_COO	fr_ketone

BCUT2D_LOGPLOW	SlogP_VSA2	fr_Al_OH	fr_ketone_Topliss
BCUT2D_MRHI	SlogP_VSA3	fr_Al_OH_noTert	fr_lactam
BCUT2D_MRLOW	SlogP_VSA4	fr_ArN	fr_lactone
BalabanJ	SlogP_VSA5	fr_Ar_COO	fr_methoxy
BertzCT	SlogP_VSA6	fr_Ar_N	fr_morpholine
Chi0	SlogP_VSA7	fr_Ar_NH	fr_nitrile
Chi0n	SlogP_VSA8	fr_Ar_OH	fr_nitro
Chi0v	SlogP_VSA9	fr_COO	fr_nitro_arom
Chi1	TPSA	fr_COO2	fr_nitro_arom_nonortho
Chi1n	EState_VSA1	fr_C_O	fr_nitroso
Chi1v	EState_VSA10	fr_C_O_noCOO	fr_oxazole
Chi2n	EState_VSA11	fr_C_S	fr_oxime
Chi2v	EState_VSA2	fr_HOCCN	fr_para_hydroxylation
Chi3n	EState_VSA3	fr_Imine	fr_phenol
Chi3v	EState_VSA4	fr_NH0	fr_phenol_noOrthoHbond
Chi4n	EState_VSA5	fr_NH1	fr_phos_acid
Chi4v	EState_VSA6	fr_NH2	fr_phos_ester
HallKierAlpha	EState_VSA7	fr_N_O	fr_piperdine
Ipc	EState_VSA8	fr_Ndealkylation1	fr_piperzine
Kappa1	EState_VSA9	fr_Ndealkylation2	fr_priamide
Kappa2	VSA_EState1	fr_Nhpyrrole	fr_prisulfonamd
Kappa3	VSA_EState10	fr_SH	fr_pyridine
LabuteASA	VSA_EState2	fr_aldehyde	fr_quatN
PEOE_VSA1	VSA_EState3	fr_alkyl_carbamate	fr_sulfide

PEOE_VSA10	VSA_EState4	fr_alkyl_halide	fr_sulfonamd
PEOE_VSA11	VSA_EState5	fr_allylic_oxid	fr_sulfone
PEOE_VSA12	VSA_EState6	fr_amide	fr_term_acetylene
PEOE_VSA13	VSA_EState7	fr_amidine	fr_tetrazole
PEOE_VSA14	VSA_EState8	fr_aniline	fr_thiazole
			fr_thiocyan
			fr_thiophene
			fr_unbrch_alkane
			fr_urea